В.П. ДЬЯКОНОВ

ПРИМЕНЕНИЕ ПЕРСОНАЛЬНЫХ ЭВМ И ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ БЕЙСИК

«Радио и связь»

В.П. ДЬЯКОНОВ

ПРИМЕНЕНИЕ ПЕРСОНАЛЬНЫХ ЭВМ И ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ БЕЙСИК

Москва
«Радио и связь»
1989

УДК 681.322-181.4.004.14
Д93
ББК 22.18

Рецензент: д-р техн. наук, проф. И. М. ВИТЕНБЕРГ

Редакция литературы по вычислительной технике

Дьяконов В. П.

Д93 Применение персональных ЭВМ и программирование на языке Бейсик. – М.: Радио и связь, 1989. – 288 с.: ил.

ISBN 5-256-00343-7

Описаны версии языка Бейсик современных персональных ЭВМ (ДВК-2М, ДВК-3, Электроника ДЗ-28, ЕС-1840, Искра-1030, IBM PC, MSX, ZX-Spectrum и др.). Даны основы программирования на Бейсике. Особое внимание уделено описанию его расширенных возможностей: синтеза звуков, создания цветных и динамических изображений, построения фигур на плоскости и в аксонометрии, картин абстрактной графики, мозаик, калейдоскопов и др. Приведено около 160 программ демонстрации возможностей ПЭВМ. Даны рекомендации по эксплуатации ПЭВМ и их периферийного оборудования.

Для инженерно-технических работников – непрофессиональных пользователей ПЭВМ; может быть полезна студентам вузов и учащимся техникумов.

Д $\frac{2404040000\text{-}103}{046\,(01)\text{-}89}$ 150-89 **ББК 22.18**

ISBN 5-256-00343-7

© Издательство "Радио и связь", 1989

ПРЕДИСЛОВИЕ

Персональные ЭВМ (ПЭВМ) – новое поколение вычислительных средств, мощный фактор ускорения научно-технического прогресса. Сейчас у нас в стране парк ПЭВМ (отечественных и зарубежных) быстро растет. Освоено серийное производство бытовых, школьных и профессиональных ПЭВМ. Интенсивно развивается их программное обеспечение. Однако разнообразие находящихся в пользовании ПЭВМ и версий Бейсика – основного языка программирования ПЭВМ – затрудняет эффективное их применение. До сих пор мало отечественной литературы по прикладному программному обеспечению ПЭВМ и их конкретному применению. Поэтому широко практикуется перевод программ с одной версии Бейсика на другую, затрудненный отсутствием описаний многих его современных версий. Описанные в литературе [1,16 – 18] версии Бейсика, кроме версии ПЭВМ PC фирмы IBM [53, 57], недостаточно учитывают возможности ПЭВМ.

Данная книга частично устраняет этот пробел. В ней параллельно рассматривается ряд версий языка Бейсик для различных отечественных и зарубежных ПЭВМ (Электроника Д3-28, ДВК-2М, ДВК-3, ЕС-1840, ZX-Spectrum, QL, MSX, IBM PC и др.). Описаны команды, позволяющие реализовать на Бейсике структурное программирование, сложные функции обработки символьной информации, управление различным периферийным оборудованием. Особое внимание уделено цветной графике ПЭВМ: заданию точек, отрезков, прямых и дуг, построению окружностей и эллипсов, изменению шрифтов, созданию графических элементов (графем) и образов (спрайтов). Рассматриваются команды динамической графики, позволяющие получать подвижные изображения (машинные фильмы), и основные приемы построения сложных фигур на плоскости и в аксонометрии. Описаны команды синтеза звуков.

Поскольку области применения ПЭВМ весьма обширны, книга не претендует на их исчерпывающее описание. Она отражает лишь личный опыт применения ПЭВМ автором при решении ряда научно-технических и инженерных задач. В отличие от ранее изданной книги автора [1] главное внимание уделено совместному использованию вычислительных и графических возможностей ПЭВМ.

В книге приведено около 160 программ, отлаженных на различных ПЭВМ. Это и простые программы, иллюстрирующие различные приемы программирования, и достаточно сложные завершенные программы построения графиков произвольных функций, их аппроксимации, расчета и моделирования топологически заданных линейных и нелинейных схем, спектрального анализа и др. В этих программах предусмотрен вывод результатов в числовой и графической форме.

Глава 1
ОБЩИЕ СВЕДЕНИЯ О ПЕРСОНАЛЬНЫХ ЭВМ

1.1. НАЗНАЧЕНИЕ

Персональные ЭВМ в отличие от "больших" ЭВМ и даже мини-ЭВМ ориентированы на работу с одним пользователем. Поэтому иногда их называют ЭВМ индивидуального пользования. Работа пользователя с ПЭВМ происходит в диалоговом режиме, имеется возможность вывода самой разнообразной информации, числовой, табличной и графической.

Современные ПЭВМ способны синтезировать сложные звуки, в том числе и человеческую речь. Уже разработаны ПЭВМ, управляемые голосом. Непрерывно улучшаются графические возможности ПЭВМ. Практически все современные ПЭВМ способны создавать сложные, в том числе динамические многоцветные изображения, что позволяет использовать их для создания машинных фильмов, сложных игр, для разработки обучающих программ.

Связь ПЭВМ с крупными вычислительными центрами обеспечивает пользователю практически неограниченный доступ к любой информации. Он может затребовать у такого центра справочные данные, необходимые для своей работы, каталоги различных изделий, нужные программы, справки о спортивных и зрелищных мероприятиях и т.д.

Ученые, инженеры и студенты могут применять ПЭВМ для научно-технических и инженерных расчетов, математического моделирования сложных объектов и систем. ПЭВМ широко применяются для управления промышленными роботами, контрольно-измерительными комплексами и станками с программным управлением. Велика роль ПЭВМ в образовании.

Персональные ЭВМ – технические устройства подлинно массового применения, поэтому они должны отвечать ряду требований, главные из них:

ориентация на работу с одним пользователем в режиме диалога (интерактивный режим);

низкая стоимость, позволяющая приобретать ПЭВМ отдельным пользователям или небольшим организациям;

высокая надежность, технологичность и ремонтопригодность, простота эксплуатации;

малые габаритные размеры и масса, обеспечивающие размещение ПЭВМ на рабочем столе (или в портфеле и даже в кармане);

широкие функциональные возможности, в частности программирование хотя бы на одном встроенном в систему языке высокого уровня (обычно это Бейсик);

работа с обширным периферийным оборудованием – дисплеем, печатающими устройствами (принтерами), графопостроителями, планшетами для ввода графической информации, телефонными акустическими модемами и др.;

ориентация на решение самого широкого круга задач.

Современные ПЭВМ в значительной мере удовлетворяют всем этим требованиям. Они постепенно становятся столь же привычным предметом обихода, как телефон и телевизор.

1.2. СТРУКТУРА И СОСТАВ ОБОРУДОВАНИЯ

Основными узлами ПЭВМ являются микропроцессор, оперативное запоминающее устройство, постоянное запоминающее устройство и средства для подключения периферийного оборудования. Все эти узлы выполнены на больших интегральных микросхемах (БИС), что и обеспечивает высокую надежность ПЭВМ, малые габаритные размеры, массу и стоимость.

Микропроцессор (МП) – устройство для выполнения основных логических и арифметических операций. МП обменивается информацией с остальными узлами ПЭВМ с помощью трех шин: адресов, данных и управления. По шине адресов МП передает информацию о том, какое из устройств в данный момент работает с ним (т.е. адрес устройства). По двухсторонней шине данных передаются данные от МП к другим устройствам или от них к МП. Наконец, шина управления осуществляет передачу сигналов управления, обеспечивая работу всех устройств в определенном порядке.

Важным параметром МП является разрядность шины данных (или просто разрядность МП). Простые ПЭВМ имеют 8-разрядные микропроцессоры (например, К580, 8080, Z80), более сложные – 16- и даже 32-разрядные. Чем выше разрядность МП, тем производительнее, быстрее он работает. Скорость выполнения операций зависит также от тактовой частоты МП (от 1 до ≈ 30 МГц).

Постоянное запоминающее устройство (ПЗУ) служит для хранения микропрограмм основных операций ПЭВМ. В частности, в ПЗУ часто хранится интерпретатор языка высокого уровня, например Бейсика. Информация в ПЗУ хранится "вечно". Однако у многих ПЭВМ ПЗУ может заменяться, например, с целью смены языков программирования, переориентации ПЭВМ на решение новых задач и т.д. Возможно подключение внешних ПЗУ.

Оперативное запоминающее устройство (ОЗУ) служит для хранения различной оперативной (т.е. сменяемой по ходу работы) информации. Это могут быть программы, массивы чисел, коды текстов и т.д. В отличие от ПЗУ содержимое ОЗУ меняется.

Важным параметром МП является максимальный объем адресуемой памяти. О нем можно судить по максимальному адресу, зависящему от числа разрядов M шины адресов. Обычно максимальный адрес равен $2^M - 1$, где вычитание 1 связано с тем, что адреса принято начинать с 0, например МП с 16-разрядной шиной адресов имеет максимальный адрес $2^{16} - 1 = 65\,535$.

К МП могут подключаться десятки и даже сотни разнообразных внешних устройств (некоторые из них указаны ниже).

Для расширения функциональных возможностей ПЭВМ используется периферийное оборудование – дополнительные блоки ПЗУ и ОЗУ, внешние накопители информации и другие устройства, которые подключаются к ПЭВМ либо прямо через каналы ввода-вывода, либо с помощью специальных контроллеров и интерфейсов.

Контроллер – устройство управления внешними устройствами ПЭВМ.

Интерфейс – совокупность аппаратных и программных средств, обеспечивающих взаимодействие периферийных устройств с ПЭВМ.

Для ручного ввода информации в ПЭВМ и управления ею служит *клавишный пульт*. Расположение большинства клавиш пульта ПЭВМ обычно такое же, как у пишущих машинок. Каждый символ, вводимый с пульта, имеет код. Коды гостированы (например, КОИ-8 в СССР или ASCII за рубежом). У отдельных ПЭВМ имеются расхождения с кодами, рекомендуемыми стандартами. Так, если у ПЭВМ IBM PC десятичный код буквы A по ASCII есть число 65, то у ПЭВМ Apple-II этот код есть число

193. Часто с помощью одной-двух клавиш можно вводить ключевые слова Бейсика. В интерфейсе клавишного пульта предусмотрены меры, предотвращающие нестабильность ("дребезг") срабатывания клавиш.

Дисплей – устройство отображения информации, чаще всего на основе электронно-лучевой трубки (ЭЛТ) или жидкокристаллический. В качестве дисплея бытовых ПЭВМ широко используются телевизоры (ТВ), черно-белые или цветные. Знаки формируются в виде матрицы (7×5, 8×8 и т.д.). Разрешающая способность дисплеев в графическом режиме от 250×250 до 1000×1000 элементов изображения.

Самый распространенный накопитель информации – кассетный магнитофон (КМ). На кассете МК-60 или МК-90 помещается до 0,4 – 2 Мбайт информации. Информация передается в виде файла, содержащего установочный сигнал частотой $f \approx 1$ кГц и длительностью 0,5 – 3 с, коды заголовка и данных. Установочный сигнал обеспечивает настройку системы автоматической регулировки уровня (АРУ) магнитофона. При считывании (воспроизведении) установочный сигнал служит для быстрого поиска начала программы. Выделенные из заголовка файла коды используются для отображения на экране дисплея названия файла, его типа, длины и контроля правильности записи (например, по совпадению суммы кодов в заголовке файла и в его поле).

Приведем данные кассетного магнитофона RQ-8100, специально предназначенного для работы с бытовыми ПЭВМ: полоса записываемых и воспроизводимых частот 100–8000 Гц, скорость движения ленты 4,8 см/с± 2 %, число дорожек – 2, выходная мощность 0,7 Вт, входное напряжение (на микрофонном входе) 0,25 мВ, время быстрой перемотки 90 с (для кассеты МК-60), питание – 4 элемента типа А316, габаритные размеры 11,9×4×20,2 см, масса 0,6 кг.

Принципиальный недостаток кассетного магнитофона – большое время поиска информации, ее записи и считывания. В профессиональных кассетных накопителях этот недостаток частично устраняется увеличением числа дорожек на ленте от 2 до 7–10 и выделением системы ускоренного поиска файлов. Ограниченное применение нашли магнитофоны с кассетой типа catridge – с "бесконечной" лентой в виде петли. Время полной загрузки программ в ОЗУ составляет несколько секунд (против нескольких минут у обычного кассетного магнитофона).

Наиболее удобен ввиду малого времени поиска, записи и считывания информации (доли секунды) *накопитель на магнитных дисках* (НМД). Информация записывается на гибкий или жесткий (типа винчестер) магнитный диск. Гибкие диски сменные, заключены в защитный чехол или конверт, который вставляется в щель накопителя, после чего диск захватывается цангой и вращается электромотором со скоростью около 300 об/мин. Запись и считывание производится магнитной головкой, которая быстро (по команде с ПЭВМ) перемещается в радиальном направлении и устанавливается на заданную дорожку. На обеих сторонах диска их может быть до нескольких десятков.

Широко применяются диски диаметром 133 мм, на них помещается до 500–1000 Кбайт информации; явна тенденция к уменьшению их диаметра (вдвое и больше). Жесткие диски несменные, но на них можно поместить до 10–500 Мбайт информации. Один или два НМД часто встраиваются в ПЭВМ. Оптические диски с лазерным считыванием позволяют накапливать на 2–3 порядка больше информации, чем магнитные.

Принтер (П) – печатающее устройство для создания копий программ, данных и результатов вычислений. Печать осуществляется печатающей головкой (игольчатой, шаровой, термической и т.д.) на обычной или специальной бумаге. В принтер входит управляемый от ПЭВМ знакогенератор, позволяющий реализовать различные типы шрифтов, и буферное ОЗУ, обеспечивающее вывод информации на печать в то время,

когда ПЭВМ перешла к выполнению новых инструкций. Скорость печати до 100–200 символов в секунду.

Для вывода высококачественных графических изображений применяют *графопостроители* (ГП). Они создают изображение не по точкам, а плавными линиями, получаемыми перемещением рисующей головки, выполненной в виде шариковой ручки или ручки с чернилами. Графопостроители, имеющие автоматически сменные головки с разным цветом чернил или пасты, позволяют создавать цветные графики. Разрешающая способность графопостроителей высокая – 0,025 – 0,1 мм, погрешность задания координат 0,2 – 1 % и скорость построения графиков 2 – 50 см/с. Качество построения графиков у них выше, чем у принтеров.

Персональные ЭВМ могут использоваться для обработки графической информации, вводимой прямо с графика или рисунка. Для преобразования координат точек графика в сигналы, вводимые в ПЭВМ, применяют специальные *графические планшеты.* Водя по графику на планшете небольшой палочкой-датчиком, можно вводить данные о графике в ПЭВМ.

Для определения координат точки графика на экране дисплея служит *световой карандаш,* с помощью которого пользователь может задать точку, прямую, кривую линию, их цвет, стирать линии и редактировать изображения – рисовать график на экране дисплея.

"Мышь" – так называют небольшую, действительно похожую на мышку коробочку, снабженную поворотными датчиками, реагирующими на перемещение ее по любой плоской поверхности (например, стола). Датчики вырабатывают сигналы, которые при нажатии специальных кнопок передаются в ПЭВМ и используются для управления движением на экране дисплея курсора, который повторяет путь "мыши" в том или ином масштабе, указывает на какое-либо изображение и т. д. Созданы ПЭВМ, например Lisa и Macintosh фирмы Apple (США) [10], вся идеология управления которыми базируется на применении "мыши". У таких ПЭВМ может не быть привычного клавишного пульта. "Меню" команд ПЭВМ появляется на экране дисплея. Нужную группу команд или отдельную команду выбирают, указав на нее маркером, передвигаемым с помощью "мыши". "Мышь" используется и для ввода в ПЭВМ графической информации.

Многие ПЭВМ могут создавать различные звуковые сигналы. Для их воспроизведения служит *громкоговоритель,* он часто встраивается прямо в ПЭВМ. Зная ноты и особенности программирования звуковых эффектов, можно писать "машинную музыку", создавать различные звуковые эффекты: щелчок клавиши, звук выстрела или грома и даже имитировать человеческую речь.

Для связи ПЭВМ с вычислительными центрами через обычные телефонные линии служат *акустические модемы* (АМ). Они содержат звуковой излучатель, посылающий сигнал в микрофон телефонной трубки и микрофон, воспринимающий звук трубки, имеют акустический контакт с телефонной трубкой, снятой с телефонного аппарата. В состав АМ входят также усилители и преобразователи аналоговых сигналов в цифровые.

1.3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

К основным техническим характеристикам ПЭВМ относятся: разрядность или тип микропроцессора, емкость ПЗУ и ОЗУ, число строк и знаков в строке, индицируемых дисплеем, габаритные размеры, масса и набор периферийных устройств. Эти характеристики зависят от назначения ПЭВМ и их конструктивного исполнения. В зависимости от последнего можно выделить ряд характерных групп ПЭВМ.

Карманные ПЭВМ (Pocket Computers), включая микрокалькуляторы, программируемые на языке Бейсик, имеют весьма малые габаритные размеры и массу. Все они снабжены жидкокристаллическим индикатором (ЖКИ) на 1–8 строк и 12–26 знаков в строке. Автономное питание от батарей и очень малая (доли милливатт) потребляемая мощность позволяют эксплуатировать такие ПЭВМ в течение 1–2 лет, не меняя батареи, храня данные и программы при отключенном питании (но оно остается подключенным к ОЗУ).

Как видно из табл. 1.1, современные карманные ПЭВМ имеют ПЗУ и ОЗУ довольно большой емкости, поэтому их можно использовать для решения достаточно сложных расчетных и экономических задач. Некоторые такие ПЭВМ имеют встроенные накопители на микрокассетах и принтеры или могут подключаться как внешние через специальные разъемы. Нередко карманные ПЭВМ содержат большое число встроенных в ПЗУ прикладных программ (статистических расчетов, вычисления специальных функций, обработки таблиц и т.д.). Поэтому емкость ПЗУ может достигать сотен килобайт.

Портативные ПЭВМ (Portable Computers) обычно имеют вид небольшой пишущей машинки, легко помещаются в портфеле. Некоторые модели имеют ОЗУ емкостью 512–1024 Кбайт, встроенные НМД и обширный набор периферийного оборудования; в отдельных моделях предусмотрен встроенный принтер. ЖКИ таких ПЭВМ с большим числом элементов – до 512X256 и выше – позволяет отображать до 16–24 строки до 80 знаков в строке, а также получать монохромные графики с высоким разрешением. ПЭВМ этого типа могут быть подключены и к цветным дисплеям. Питаются они и от сети и от батарей (в последнем случае время непрерывной работы составляет от 8 до 24 ч.) В настоящее время ПЭВМ этого класса интенсивно развиваются (табл. 1.2).

Бытовые или *домашние* (Home Computers) *ПЭВМ*, часто выполненные в виде клавишного пульта, подключаются к внешнему дисплею или цветному телевизору, кассетному магнитофону (или НМД) и принтеру. В табл. 1.3 приведены данные одной из характерных, наиболее распространенных и дешевых ПЭВМ этого класса ZX-Spectrum^{+} (ее характеристики, за исключением габаритных размеров, массы и числа клавиш, аналогичны другой массовой модели ZX-Spectrum). Модель ZX-Spectrum^{+} имеет 8-цветную графику.

Таблица 1.1

Характеристики карманных ПЭВМ

Характеристика	Электроника МК-85 (СССР)	PC-1245	PC-1500	PC-1600	HP-71B (Hewlett-Packard, США)
		(Sharp, Япония)			
Разрядность МП	16	8	8	8	4
Емкость, Кбайт:					
ПЗУ	16	25	16	96	64/320
ОЗУ	2	2,2	3,5/19,5	16/80	16/33,5
Число строк/знаков ЖКИ	1/12	1/16	1/26	4/25	1/22
Габаритные размеры, мм	165X73X13	130X57X9	195X86X22,5	195X86X22,5	190X98X13
Масса, кг	0,15	0,1	0,375	0,39	0,34
Периферийные устройства	Нет	КМ, П	КМ, П, ГП, АМ	КМ, НМД, П, ГП, АМ	КМ, НМД, П, ГП, АМ

Таблица 1.2

Характеристики портативных ПЭВМ

Характеристика	HP-110 (Hewlett-Packard, США)	5000G (Sharp, Япония)	VADEM (Osborn Computer, США)	GAVILAN (Magirus Datentechik, ФРГ)	M-10 (Olivetti, Италия)
Разрядность МП	16	16	16	16	8
Емкость, Кбайт:					
ПЗУ	384	196	16	8	32–64
ОЗУ	272	128–256	128–512	96–288	8–32
Число строк/знаков ЖКИ	16/80	8/80	16/80	16/80	8/40
Число элементов графики	480/128	640/80	480/128	480/128	240/64
Габаритные размеры, мм	320×250×70	326×305×88	312×229×121	290×290×70	300×210×50
Масса, кг	3,9	5,8	4,7	4	1,7
Встроенное ЗУ	ЗУ на БИС	ЗУ на ЦМД	НГМД	НГМД	Нет
Встроенный принтер	Нет	Нет	Нет	Есть	Нет

Таблица 1.3

Характеристики бытовых ПЭВМ

Характеристика	ZX-Spectrum+ (Sinclair Research Ltd, Великобритания)	PC XT (IBM, США)	Osborne PC (Osborne, США)	Macintosh (Apple, США)	Apricot Xi (Kwest, Великобритания)
Разрядность МП	8	16	16	32	16
Емкость, Кбайт:					
ПЗУ	16	64	–	64	–
ОЗУ	48	128–640	256–512	128	512–896
Число строк/знаков текста	22/32	25/80	24/80	25/80	25/80
Число элементов графики	256×176	320 (640) / 200	640/200	512/342	800/400
Габаритные размеры, мм	320×160×40	500×410×142	520×330×230	250×250×350	420×320×100
Масса, кг	0,4	14,5	11,9	9	6,5
Емкость НМД, Мбайт	Нет	0,36 20	2×0,36 33	0,4	0,72 10
Внешние устройства	НГМД, НМЛ, П, ТВ, Д	П, Д, ГП, АМ	П, Д, ГП, АМ и др.	"Мышь", П, Д, ГП, АМ и др.	П, Д, ГП, АМ и др.

У таких моделей наблюдается явная тенденция к увеличению емкости ОЗУ и ПЗУ (обычной становится емкость ОЗУ 128-256 Кбайт). Как правило, эти ПЭВМ выполнены на 8-разрядном МП (Z80, 8080, 8085 и др.), но стали появляться и модели с 16-разрядным МП. У последних моделей до нескольких десятков и даже тысяч цветовых оттенков, что позволяет использовать такие ПЭВМ для сложных игр.

Среди бытовых ПЭВМ за рубежом наибольшее распространение нашли модели фирм Commodore, Apple и Atari (США). Некоторые из них по основным характеристикам не уступают профессиональным стационарным ПЭВМ. Например, модель Atari ST, построенная на 16-разрядном МП 68000, имеет ОЗУ емкостью 512 Кбайт (с возможностью адресации до 16 Мбайт), ПЗУ емкостью 192 Кбайт, 16-цветную графику на 320×200 элементов и 640×400 элементов в монохромном режиме. Некоторые модели таких ПЭВМ имеют дополнительные МП (сопроцессоры) для выполнения быстрых арифметических операций, создания сложных видео- и звуковых сигналов.

Стационарные ПЭВМ делятся на собственно персональные и *профессиональные* (с улучшенными техническими характеристиками). Среди этих ПЭВМ своего рода стандартом стали массовые модели фирмы IBM, например PC XT. Совместимость по программному обеспечению с этими ПЭВМ рассматривается как необходимое качество других моделей. Однако имеются и исключения из этого правила, например фирма Apple (США) выпускает ПЭВМ Macintosh, принципиально несовместимые с ПЭВМ IBM PC. Характеристики ряда зарубежных моделей стационарных ПЭВМ приведены в табл. 1.3.

Как видно из табл. 1.3, масса и габаритные размеры стационарных ПЭВМ сравнительно невелики и их можно легко переносить с места на место. Базовый комплект ПЭВМ – системный блок, клавишный пульт, дисплей и принтер – занимают часть письменного стола. В системный блок обычно встроены один-два НМД.

Обширная номенклатура стационарных ПЭВМ выпускается в СССР (табл. 1.4), последние модели их (ЕС-1840, ЕС-1841, Искра-1030.11) по основным характеристикам и программному обеспечению совместимы с моделями IBM PC. Модель ПЭВМ "Агат" совместима с моделью Apple-II.

1.4. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПЭВМ

Технические характеристики ПЭВМ оказывают заметное влияние на предельную сложность решаемых задач. Эффективность применения ПЭВМ не столько зависит от технических характеристик ПЭВМ, сколько от их программного обеспечения. Наличие хорошего программного обеспечения значительно облегчает работу с ПЭВМ. Программное обеспечение ПЭВМ обычно подразделяется на системное и прикладное.

Системное программное обеспечение обеспечивает правильное взаимодействие всех узлов вычислительной системы. В него входят машинно-ориентированные языки и программа управления системой и контроля за ее состоянием (монитор). Совокупность системных программ управления вычислительной системой обычно называется операционной системой (ОС). Если последние расположены на магнитном диске, их называют дисковой операционной системой (ДОС). ОС и ДОС резко расширяют возможности ПЭВМ. Современные ПЭВМ, в сущности, воспринимают и обрабатывают лишь двоичные числа (с основанием 2). Ими кодируется вводимая в ПЭВМ информация и команды, исполняемые МП. Однако такие числа неудобны для людей-пользователей ПЭВМ. Для компактного представления чисел в двоичных разрядах они объединяются, образуя байт-единицу, имеющую $2^8 = 256$ различных значений. Этого вполне достаточно для кодирования всех букв одного-двух алфавитов (например, латинского и русского), спецзнаков и команд ПЭВМ. Поэтому коды команд ПЭВМ задаются байтами. У многих ПЭВМ для этого можно использовать и при-

Таблица 1.4

Характеристики отечественных ПЭВМ и вычислительных комплексов

Характеристика	Агат	Искра-226	ДВК-2М	ДВК-3	ДВК-4	Искра-1030.11	ЕС-1840 (ЕС-1841)	Электроника		
								85	ДЗ-28	ТЗ-29МК
Емкость ОЗУ, Кбайт (мин/макс)	63/256	112/176	56	64/248	64/248	256/512	356/540	512	128	128/256
Размер экрана дисплея, мм	350*	310*	220×160	205×130	220×160	400	250×155	220×160	220×160	220×140
Число строк/символов текста	32/32, 32/64	24/80	24/80	24/80	24/80	25/40 25/80	25/80	25/80	24/80	24/80
Число элементов графики	64×64 128/128 256/256	560×256	Нет	440×280	256×256	320×200 640/200	640/200	960/240	Нет	512/256
Число цветов	16	2	2	2	8	8	2	2	2	2
Встроенный накопитель	НГМД	КН	Нет	НГМД	НГМД	НГМД	НГМД	НГМД НЖМД	КН	КН
Емкость внешней памяти (накопителя), Кбайт	2×0,125	80	2×220 (2×800)	2×400 (2×800)	2×320	2×320 (10.10^3)	2×320 (10.10^3)	2×400 (10.10^3)	400	144
Внешние устройства	П	П, ГП, НМД	П, ГП, НГМД	П, ГП,	П, ГП,	П, ГП,	П, ГП,	П, ГП,	П, ГП НМГД	П, ГП НМГД

*Указан размер экрана по диагонали.

вычные десятичные числа. Перевод восьми разрядов ($X_0 - X_7$) байта в десятичное число K_{10} выполняется по формуле

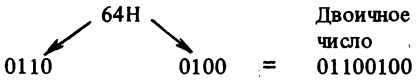
$$K_{10} = X_7 \cdot 2^7 + X_6 \cdot 2^6 + \ldots + X_1 \cdot 2^1 + X_0 \cdot 2^0.$$

Например, если $K_{10} = 100$, то

$$100 = 0 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0.$$

В двоичной форме число 100 имеет вид 01100100, где 0 и 1 (логические нуль и единица) – разряды двоичного числа.

Еще более компактную форму имеют шестнадцатеричные числа с основанием 16. Первые 16 таких чисел задаются следующим образом: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Чтобы не путать их с десятичными числами, после них обычно ставится специальный знак – обычно буква H или h. Так AH или Ah есть шестнадцатеричное число A, эквивалентное десятичному числу 10. Шестнадцатеричные числа легко переводятся поразрядно в двоичные числа, например:



Шестнадцатеричные числа более компактны. Так, десятичное число 100 в шестнадцатеричном виде есть 64 – буква H может опускаться, если заранее известно, что речь идет о шестнадцатеричных числах. По этой причине адреса в ОЗУ и ПЗУ удобно указывать шестнадцатеричными числами.

Машинно-ориентированные языки – это языки, команды которых задаются кодами (двоичными, десятичными или шестнадцатеричными числами). Такие языки открывают полный доступ ко всем ресурсам ПЭВМ и позволяют наиболее полно и оперативно пользоваться всеми ее возможностями. Однако эти языки крайне неудобны для пользователя и ненаглядны. По существу, это просто списки чисел – машинных кодов и адресов.

Ассемблером ПЭВМ называют специальный символический язык команд – определенным машинным кодам однозначно соответствует некоторый мнемокод обычно в виде сокращенных английских слов. Например, у широкораспространенных ПЭВМ на базе МП 8080 и Z-80 операция загрузки числа N в регистр, принимающий данные, имеет десятичный код 62. На ассемблере она обозначается как LDA, (load – загрузить). Программа, переводящая мнемокоды в машинные коды, также называется ассемблером. А обратный перевод делается с помощью программы-дисассемблера. Число команд языка ассемблера велико – обычно несколько сотен. Эти команды могут быть одно- и многобайтовыми (обычно 2–3 байт). Ассемблер полезно знать опытным пользователям и программистам: с его помощью можно использовать ПЭВМ с максимальной эффективностью. Список команд ассемблера почти всегда приводится в описании конкретных ПЭВМ или применяемых в них микропроцессорах.

Монитор – специальная программа, обеспечивающая управление с клавишного пульта ПЭВМ, загрузку в нее других языков программирования, работу с ассемблером, диагностику ПЭВМ и т.д. Обычно эта небольшая программа, задающая и организующая начальный диалог пользователя с ПЭВМ, входит в состав ОС или ДОС.

Большинство пользователей-непрофессионалов предпочитает использовать *языки высокого уровня*. Все они содержат более или менее полный набор инструкций в виде слов и спецзнаков, понятных пользователю и заменяющих сразу ряд инструкций в машинных кодах. Ниже указаны основные из таких языков.

Компиляторы и интерпретаторы – это специальные программы, переводящие инструкции языков высокого уровня в машинные коды. Компиляторы воспринимают

всю программу, вводимую в ПЭВМ, и разом переводят (транслируют) ее в совокупность машинных кодов. Это обеспечивает высокую скорость проведения вычислений по оттранслированной программе. Однако любые исправления в программе требуют ее повторной трансляции, которая занимает определенное (нередко не малое) время.

Интерпретаторы воспринимают каждую инструкцию по мере ее появления и преобразуют последнюю в машинный код. Это замедляет вычисления, но облегчает диалог пользователя с ПЭВМ: любую команду можно выполнить сразу после ввода.

Проблемно-ориентированные языки высокого уровня предназначены для решения определенного (нередко весьма широкого) круга задач. Их инструкции задаются словами и заменяют множество машинных команд. Запись математических выражений на таких языках близка к естественной форме. Наиболее известны такие языки, как Алгол, Фортран, Бейсик, Паскаль, Форт, Кобол и др.

Бейсик (Beginner's Allpurpose Symbolic Instruction Code многоцелевой язык инструкций для начинающих) – самый массовый и популярный язык программирования ПЭВМ. Бейсик, к сожалению, имеет множество версий. Сейчас ведется работа по его стандартизации [19, 20], в частности, в основу стандарта этого языка для учебных ПЭВМ взяты версии ПЭВМ IBM PC (США) и MSX (фирма "Ямаха", Япония).

Невысокая скорость счета, присущая обычным версиям Бейсика (интерпретирующим), является определенным недостатком, который преодолевается созданием компиляторов Бейсика. Другой его недостаток – неприспособленность к структурному программированию. У ряда его современных версий он полностью устранен. Нередко Бейсик является языком операционной системы ПЭВМ, обычно он встроен в ПЗУ, и его загрузка занимает доли секунды.

Прикладные программы – наиболее массовый тип программ. Некоторые ПЭВМ имеют библиотеки из многих тысяч таких программ. Часто используются прикладные программы для задания и обработки таблиц, построения графиков и рисунков, редактирования текстов, выполнения математических, инженерных и экономических расчетов, обучения и игр. Пользователь ПЭВМ должен уметь грамотно составлять свои собственные прикладные программы и пользоваться готовыми программами.

Глава 2

ОСНОВЫ ПРОГРАММИРОВАНИЯ РАСЧЕТНЫХ ОПЕРАЦИЙ НА ЯЗЫКЕ БЕЙСИК

2.1. АЛФАВИТ ЯЗЫКА БЕЙСИК, СТРУКТУРА ПРОГРАММ И ВИДЫ ПЕРЕМЕННЫХ

Программа на языке Бейсик есть запись последовательности сгруппированных в пронумерованные строки команд (инструкций), под действием которых ПЭВМ выполняет необходимые для решения заданной задачи операции. Команды представляют собой совокупность различных символов – цифр, букв и специальных знаков, перечисленных ниже.

Алфавит Бейсика включает 26 латинских прописных *букв*:

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.

В расширенных версиях Бейсика в алфавит входит столько же строчных латинских букв:

a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z.

У отечественных ПЭВМ алфавит Бейсика расширяется включением в него строчных и прописных букв русского алфавита (от А до Я и а . . .я).

Цифры – обычные десять арабских цифр:

Ø, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Нуль в программах на Бейсике обычно перечеркивается наклонной чертой, чтобы нельзя было спутать его с буквой О.

Знаки арифметических операций в Бейсике следующие: – вычитание или присвоение знака минус, + сложение, * умножение, / деление, ↑ или ¬ – возведение в степень.

Специальные знаки (основные):

( ) – скобки	! – восклицательный знак
: – двоеточие	= – равенство
; – точка с запятой	< – меньше
. – точка	> – больше
, – запятая	\ – наклонная черта
" " – кавычки	# – "решетка"
' – апостроф	☼ – "чертик" или "солнышко"
? – вопросительный знак	$ – знак денежной единицы
& – амперсанд	% – знак процентов
@ – "относительно"	

Наряду с основными и общепринятыми специальными знаками в расширенных версиях Бейсика могут использоваться и другие знаки, например, ≥, ≤, ≠, π, # PI или PI, ! – знак вычисления факториала.

Строки бейсик-программ нумеруются в последовательном порядке – от меньшего номера к большему. Так они и выполняются, если в программе нет специальных указаний на иной порядок выполнения. Рекомендуется нумеровать строки числами, кратными 5 или 10. Это позволяет при необходимости вставлять в промежутках между основными строками вспомогательные, необходимые в ходе модификации, редактирования или отладки программ. Максимальный номер строки в зависимости от вида ПЭВМ обычно от 7500–10 000 до 500 000. Некоторые современные ПЭВМ имеют специальные средства для автоматической нумерации или перенумерации строк с заданным номером начальной строки и заданным шагом изменения номеров последующих строк.

Команды языка Бейсик подразделяются на директивы, операторы и функции. Они записываются в виде английских слов (иногда сокращенных) прописными латинскими буквами. Лишние пробелы обычно игнорируются, однако в словах со слитными буквами их лучше не вводить.

Директива выполняется непосредственно после ввода и служит для изменения программ или файлов. Нередко директивы могут включаться в текст программ. Однако многие из них останавливают программу, обнуляют переменные или делают их неопределенными, ведут к уничтожению программы. Например, директива NEW (обновление) в ряде версий Бейсика стирает данные и программу и очищает ОЗУ, обычно используется вне

программы перед ее вводом. Однако, если поместить директиву NEW в конец программы, она будет исполнена один раз и затем самоуничтожится.

Оператор выполняется, как только он встречается в программе. Для расширения возможностей языка Бейсик предусмотрены дополнения к операторам – *модификаторы.* Введение в состав операторов модификаторов повышает логические возможности программирования, делает программы более наглядными, гибкими и короткими.

Функции обеспечивают выполнение определенных функциональных преобразований (например, вычисление логарифма или синуса переменной x).

Нередко различия между директивами, операторами и функциями специально не оговариваются. В этом случае все команды, если они выполняются по программе, называют операторами.

Таким образом, программа на языке Бейсик представляет собой совокупность команд (директив, операторов и функций), разбитых на отдельные предложения, помещаемые вслед за номером строки. Программа обычно завершается директивой END (конец). Однако у многих ПЭВМ она отсутствует, и признаком завершения программы является последняя команда в строке с наибольшим номером. Для останова программы в любом месте служит директива STOP. При ее исполнении ПЭВМ останавливает счет и выводит на индикацию сообщение об останове с указанием номера строки, например

```
: ОСТАНОВ В СТРОКЕ 10Ø
```

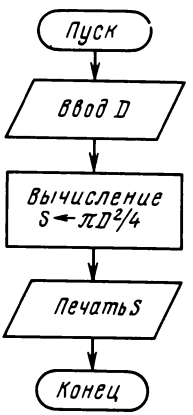
(Более подробные данные о директивах STOP и END даны в § 6.1.)

Структура программ может быть линейной, разветвляющейся, циклической и комбинированной. В программе с линейной структурой все команды выполняются в строго определенном порядке от первой строки к последней.

Примером программы с линейной структурой является программа вычисления площади круга по формуле $S = \pi D^2/4$. Алгоритм, т. е. указание последовательности вычислений, в этом случае следующий.

1. Задаем диаметр окружности D.
2. Вычисляем $\pi D^2/4$ и придаем его числовое значение переменной S.
3. Задаем печать значения S.

Схема алгоритма и программа, его реализующая, приведены на рис. 2.1.




```
10 INPUT D
20 LET S=#PI*D*D/4
30 PRINT S
 3.141592654

10 INPUT D:LET S=#PI*D*D/4:PRINT S
 3.141592654
```

Рис. 2.1

В приведенной программе в каждой из трех строк стоит только один оператор. В строке 10 оператором INPUT D задается ввод значения D. При исполнении этого оператора ПЭВМ печатает знак вопроса (?) и ждет ввода числа. В строке 20 с помощью оператора присваивания LET переменной S придается значение $\pi D^2/4 = PI * D*/4$. Наконец, в строке 30 оператором PRINT задается печать значения S на принтере или вывод его на экран дисплея.

В одной строке может быть не одна, а несколько инструкций, разделенных знаками, обычно : или \ . Например, всю приведенную выше программу можно представить одной строкой (вторая программа на рис. 2.1).

Размещение в каждой строке по одному оператору упрощает исправления в программах, делает их более наглядными для неопытных пользователей. Однако при этом текст (листинг) сложных программ оказывается слишком длинным. Программы даже умеренной сложности в этом случае не вмещаются в одну страницу дисплея, а распечатка их на принтере требует большого расхода дефицитной и дорогостоящей бумаги. Поэтому в программах целесообразно задавать несколько инструкций в каждой строке, группируя их по виду, например директивы очистки памяти, операторы ввода, операторы и функции для вычисления, операторы и директивы вывода результатов.

Основной вид информации, которой оперирует ПЭВМ, – *числа*. Они представляются в виде последовательности соответствующих цифр с особыми знаками: знаками мантиссы и порядка + или –, знаком ввода порядка E (от слова Extent – степень) и знаком разделения целой и дробной части чисел – точкой вместо разделительной запятой.

Числовая константа – запись числа, обычно непосредственно в программу. Например, команда LET N = 5 означает, что переменной N придается числовое значение, определяемое константой 5.

Числа могут быть целыми, т. е. не содержать дробной части:

Ø, Ø1, 1, 12, – 124, 5678

Операции с целыми числами ПЭВМ в пределах своей разрядной сетки выполняет точно.

Форма представления чисел на Бейсике весьма близка к обычной математической форме записи. Дробные числа представляются в виде

3.142 17.896 5.Ø123 –123.456

Однако, если первая цифра мантиссы число 0, ее можно не вводить. Так, правильные дроби записываются в виде

Ø.Ø123 .Ø123 .ØØ9 –.256

Наиболее общей формой представления чисел на Бейсике является *десятичная форма.* Ей соответствуют десятичные числа, записываемые в виде $\pm M \cdot 10^{\pm E}$, где M – мантисса и E – порядок. Обычно порядок записывается после знака ввода порядка E, например

2.141EØ5 –3.14E2 .123E –12
(214100) (–0,314) $(0{,}123 \cdot 10^{-12})$

Внизу в скобках дано обычное обозначение соответствующих чисел. Подобная форма записи называется *показательной* (иногда экспоненциальной).

Большинство ПЭВМ при вычислениях осуществляют *нормализацию чисел.* Мантисса числа приводится к виду, когда до десятичной точки стоит цифра от 1 до 9. К примеру, число .123E –12 после нормализации принимает вид 1.23E –13. Некоторые ПЭВМ обеспечивают выдачу чисел в наиболее удобной для специальных расчетов форме. Например, ПЭВМ для научных расчетов обычно имеют три формы представления чисел (иногда их называют форматами).

Нормальная – в виде обычного десятичного числа при его значении $>10^{-3}$ и $<10^{10}$. В этой форме число –3.14E2 в виде –0.0314. Если значение числа выходит за эти пределы, оно представляется в показательной форме.

Показательная (научная) – соответствует приведённой выше десятичной форме с нормализованной мантиссой.

Инженерная – особый вид показательной формы, при которой показатель степени равен 0 либо кратен трем. Например, число 214100 в этом случае представляется в виде 214.1E3, а число –0,0314 в виде –31.4E–6. Кратность порядка 3 полезна, если вспомнить, что на практике такую кратность изменения порядка при переходе от одной размерности к другой имеют многие физические величины (миллиграмм, грамм, килограмм или миллиом, ом, килоом).

Максимальное значение порядка десятичных чисел зависит от назначения ПЭВМ. У ПЭВМ, ориентированных на инженерные и научные расчеты, порядок может быть от –99 до +99 (при этом мантисса может содержать от 8 до 16 цифр и более). У ПЭВМ общего назначения порядок ограничен числами, например, от –39 до +38, а мантисса содержит 8 или 9 цифр – так называемые числа одинарной точности. Возможны и иные значения порядка – у языка Бейсик-80 для переменных "двойной точности" порядок лежит в пределах от –308 до +308.

Некоторые типы ПЭВМ наряду с числами одинарной точности могут оперировать и числами двойной точности. Под не вполне правильным термином "точность" обычно подразумевается рабочая разрядность чисел. Например, одинарная точность соответствует 8 цифрам представления мантиссы, а двойная – 16 цифрам, чем больше разрядность чисел, тем больше емкость ОЗУ требуется для их хранения. Таким образом, правильнее говорить об обычной и двойной разрядности чисел. Двойной разрядности соответствует меньшая погрешность вычислений и соответственно большая (но отнюдь не вдвое!) точность.

Иногда ПЭВМ имеют *скрытые разряды,* т.е. выполняют операции с числами, разрядность которых на 1 – 3 единицы больше разрядности, в которой числа выводятся на индикацию. Это позволяет резко уменьшить погрешность вычислений, например вычислять математические функции со всеми верными индицируемыми цифрами результата. Однако повышение разрядности чисел сопровождается снижением быстродействия ПЭВМ.

Переменные – величины, обозначенные определенными символами и способные по ходу вычислений принимать различные символьные или числовые значения. Символьные (или текстовые) переменные будут подробно рассмотрены в гл. 4. Поэтому ниже речь пойдет только о числовых переменных. Существует несколько типов таких переменных.

Простые переменные – переменные, обозначаемые совокупностью латинских букв и цифр. Обычно простейшие версии Бейсика допускают две формы простых переменных: обозначаемых любой прописной латинской буквой (от A до Z) либо буквой и цифрой (от Ø до 9). Таким образом, простые переменные могут иметь такой вид: A, B, X, Y, AØ, B1, X8, Y9 и т.д. В сложных версиях Бейсика вполне допустимы простые переменные, содержащие в своем наименовании несколько букв (до 2 – 10): AFTER, START, Finish, Hello. Для применения простых переменных не требуется каких-либо специальных указаний, кроме их предварительного определения. Иногда неопределенным переменным придаются нулевые значения.

Целочисленные переменные – переменные, способные принимать только целочисленные значения. Арифметические операции с такими переменными ПЭВМ выполняет точно. Переменные (и константы) объявляются целочисленными путем ввода после них знака %. Например, 5% – целочисленная константа (число 5 всегда без дробной части), A%, A9%, AFTER% – соответствующие целочисленные переменные.

Переменные и константы без специального указания вида переменных (по умолчанию) задаются как переменные и константы одинарной (обычной) точности, у ПЭВМ, допускающих двойную точность, знак E у порядка указывает на обычную точность. На то же может указывать знак !. Например, –3.141! 7.268451E–09 соответствуют константам с "одинарной точностью".

Константы или переменные "двойной точности" отмечаются знаком # после них или указанием порядка после буквы D, например:

A# B1# AFTER# 7.2684516789D–12

Индексированные переменные – переменные одно-, двух- или трехмерных массивов.

Одномерный массив – последовательность переменных с произвольными числовыми значениями вида

$a_0 \quad a_1 \quad a_2 \quad a_3 \quad \ldots \quad a_{n-1} \quad a_n$

Наряду с самими числами массив характеризуется длиной или числом элементов, а также номером начального элемента. В данном случае последний есть 0. Возможны массивы вида

$a_1 \quad a_2 \quad a_3 \quad a_4 \quad \ldots \quad a_{n-1} \quad a_n$

начальный элемент которых имеет номер (индекс) 1. Размерность массива обычно ограничена емкостью ОЗУ. Однако иногда она специально ограничивается (например, задается $i < 256$).

Перед использованием одномерного массива нужно задать имя индексированной переменной и размерность массива n. Так, приведенный массив чисел a_i можно задать в виде индексированных переменных $A(I)$, где $I = 0, 1, 2, \ldots, n$ или $I = 1, 2, \ldots, n$ (зависит от конкретной ПЭВМ).

Задание размерности массива выполняется операторами

[HC] DIM ⟨Переменная⟩ (Число n)

или

[НС] COM ⟨Переменная⟩ (Число n)

DIM – сокращение от английского слова dimmension (размер), а COM – common (общий). Например, если $n=10$ (11 чисел при начале отсчета с 0 или 10 при начале отсчета с 1), то операторы DIM A(1∅) или COM A(1∅) означают массив $A(I)$, индекс которого может принимать значения от $I=0$ или 1 до 10.

Двухмерные массивы задают матрицу чисел, например:

$$\begin{array}{cccccc} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & b_{m3} & \dots & b_{mn} \end{array}$$

Эта матрица характеризуется числом строк m и числом столбцов n. Говорят, что она имеет размер $m \times n$. Перед использованием двухмерного массива надо задать имя индексированных переменных массива и размерность массива с помощью операторов

[НС] DIM ⟨Переменная⟩ (Число m, Число n)
[НС] COM ⟨Переменная⟩ (Число m, Число n)

Например, оператор DIM B(1∅,5) задает массив под матрицу из $m=10$ и $n=5$ элементов, переменная $B(I,J)$ означает индексированные переменные двухмерного массива с индексами I и J (соответствует b_{ij}).

Трехмерный массив можно представить как p матриц размера $m \times n$ (рис. 2.2). Таким образом, трехмерный массив имеет размер $m \times n \times p$. Перед его использованием задается имя переменных и размер массива с помощью операторов

[НС] DIM ⟨Переменная⟩ (m, n, p)
[НС] COM ⟨Переменная⟩ (m, n, p)

Например, оператор DIM A% (1∅, 1∅, 1∅) задает трехмерный массив из $10 \times 10 \times 10 = 1000$ целочисленных переменных с общим именем $A\%(I, J, L)$, где индексы I, J и L могут принимать значения от 0 или 1 до m, n и p соответственно (в нашем случае $m=n=p=10$). В большинстве ПЭВМ используются только одно- и двухмерные массивы.

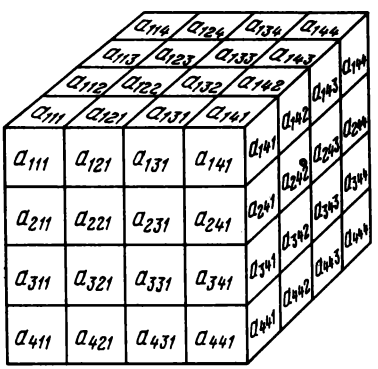


Рис. 2.2

Как правило, переменные вида A, AI (где $I=0, 1, \dots, 9$), $A(I)$, $A(I, J)$ и $A(I, J, L)$ представляют собой разные переменные. Однако у некоторых простых ПЭВМ возможно исключение из этого правила. Так, возможны случаи, когда переменные A1 и A(1), A2 и A(2) и т. д. физически равноценны, т. е. занимают одни и те же ячейки ОЗУ (ПЭВМ FX-702P). У ПЭВМ с индексацией массивов с нуля нередко переменные A(∅, ∅)

и A∅ или A оказываются физически равноценными. Эти особые случаи нужно учитывать, поскольку они могут привести к грубым и труднообнаруживаемым ошибкам в ходе вычислений.

2.2. ВЫВОД ИНФОРМАЦИИ (ОПЕРАТОРЫ REM, PRINT, SET, TAB, AT, WAIT, PAUSE)

Комментарии (словесные, формульные и др.) к программе могут выводиться с помощью оператора REM (от слова remark – комментарий или замечание). Вид этого оператора следующий:

[НС] REM ⟨Комментарий⟩

Комментарий может занимать целую строку, например

1∅ REM ПРОГРАММА ЧИСЛЕННОГО ИНТЕГРИРОВАНИЯ

может стоять в конце строки

1∅∅ LET X = ∅ : REM ОБНУЛЕНИЕ ПЕРЕМЕННОЙ X

После комментария нельзя (в одной строке с ним) ставить другие исполняемые операторы. Например, конструкция строки вида

1∅∅ REM ОБНУЛЕНИЕ ПЕРЕМЕННОЙ X : LET X = ∅

как правило, является ошибочной, поскольку фрагмент строки LET X = = ∅ будет восприниматься просто как часть комментария, а не как самостоятельная инструкция.

Комментарии не выводятся на печать (или экран дисплея) при обычном выполнении программы, индицируются только при распечатке или просмотре текста (листинга) программы. Умелое использование комментариев делает программу более наглядной. Комментарии широко применяются, например, для пояснения положенных в основу программы алгоритмов и численных методов, указания авторов и даты подготовки программы и других полезных, но не всегда обязательных сведений. Однако злоупотреблять комментариями не следует – они удлиняют листинг программы и сокращают емкость ОЗУ.

Оператор PRINT (печать) служит для вывода информации на печать или на экран дисплея. Информацией является комментарий, заключенный в апострофы или кавычки, символьное значение символьной переменной (см. гл. 4), графические знаки–графемы (см. гл. 5), числовое значение константы, переменной любого вида и арифметического выражения.

Общая формула оператора PRINT следующая:

[НС] PRINT [Знаки формата , . ;] ['Комментарий'] [, : ;] [Константа Переменная Выражение] [, . ;] и т.д.

Оператор PRINT может стоять в любом месте строки. Приведем примеры применения оператора PRINT для вывода цифробуквенной информации.

Пустой оператор PRINT (без списка переменных и указаний) вызывает перевод строки – печать пустой строки. Например, строка

```
10 PRINT : PRINT : PRINT
```

обеспечивает печать трех пустых строк, т.е. создает пробел из трех строк. Строка

```
10 PRINT 'ПРОГРАММА ЧИСЛЕННОГО ИНТЕГРИРОВАНИЯ'
```

обеспечивает печать комментария ПРОГРАММА ЧИСЛЕННОГО ИНТЕГРИРОВАНИЯ. Комментарий обычно заключается в апострофы или кавычки (зависит от типа ПЭВМ).

В состав оператора PRINT могут вводиться разделители и знаки формата. Разделителями являются обычно запятая (,) и точка с запятой (;). Запятая как разделитель обеспечивает печать каждого элемента списка в определенном секторе строки. Если число элементов больше числа секторов, то непоместившиеся элементы переносятся в другую строку. Например, если в строке три сектора, то исполнение оператора

```
PRINT 'КОНСТАНТЫ', 3.141, 25, 78, 12E –5
```

приведет к распечатке вида

```
КОНСТАНТЫ                3.141          25
78                   12E –5
```

Разделитель в виде точки с запятой (;) обеспечивает печать одного элемента сразу вслед за другим. Например, строка

```
50 PRINT 'КОНСТАНТА ПИ ='; 3.141; 25; 78; 12E –5
```

при исполнении дает следующую распечатку:

```
КОНСТАНТА ПИ = 3.141257812E –5
```

У некоторых ПЭВМ разделитель (;) автоматически создает между отдельными элементами списка один или два пробела. В этом случае распечатка приведенной выше строки будет иметь вид

```
КОНСТАНТА ПИ = 3.141 25 78 12E –5
```

Аналогично оператором PRINT выводятся числовые значения переменных. Например, если A = 12, B = 34, C = 175 и D = 2.34E –15, то исполнение строки

```
20 PRINT A, B, C , D
```

дает распечатку вида (при трех секторах в строке)

```
12              34              175
2.34E –15
```

Вот пример комбинированного использования разделителей:

```
20 PRINT "A ="; A, "B ="; B, "C ="; C, "D ="; D
```

Исполнение такой строки дает следующую распечатку:

```
A = 12          B = 34          C = 175
D = 2.34E –15
```

При исполнении строки

```
2Ø  PRINT "A=" ; A : PRINT "B=" ; B : PRINT "C=" ; C:
    PRINT "D="; D
```

распечатка располагается в столбец:

```
A = 12
B = 34
C = 175
D = 2.34E  –15
```

Указатели формата выдачи числовых данных, к сожалению, различны у разных версий Бейсика. Это затрудняет дословный перевод конструкций с операторами вывода с одной версии Бейсика на другую.

У систем подготовки программ на базе микроЭВМ "Электроника Д3-28" знак #Ø после оператора PRINT указывает на вывод информации на экран дисплея, а знак #1 ее распечатку на принтере. При использовании оператора PRINT без этих знаков (по умолчанию) информация выводится на экран дисплея. Указание об устройстве, на которое выдается информация, запоминается. Таким образом, знак #Ø или #1 необходимо использовать в первый раз, в последующих операторах его можно не использовать.

Формат выдачи указывается следующим образом (n_1 – число знаков до десятичной точки, n_2 – после нее);

!Fn_1 n_2! – выводится знак числа в экспоненциальной форме, мантисса, знак порядка, символ порядка Е, две цифры порядка,

!n_1.n_2! – выводится знак числа в формате с фиксированной запятой (если число не умещается в заданном формате, выводится столько символов, сколько позиций отведено на печать),

!E! – выводится число в показательной форме с плавающей запятой, содержащее знак числа, точку, 12 цифр мантиссы, знак порядка, символ порядка Е и 2 цифры порядка.

При первом включении устанавливается формат ! F 1.9 !. В пределах значений n_2 все нули мантиссы выводятся на индикацию. Например, число 1,25 индицируется в виде 1.25ØØØØØØØ, а число 1 в виде 1.ØØØØØØØØØ. При индикации целых чисел вывод всех нулей создает неудобства. Например, переменная a_2 = 5 одномерного массива при исполнении оператора

```
PRINT ! F 1.9 !'A ('2') =' ; A(2)
```

даст такую распечатку:

```
A(2.ØØØØØØØØØ) = 5.ØØØØØØØØØ
```

Однако, задав оператор PRINT в форме

```
PRINT ! F 2.Ø!'A ('2') ='! F 1.9 ! A (2)
```

получим приемлемый вид распечатки

```
A(2) = 5.ØØØØØØØØØ
```

У некоторых ПЭВМ вывод лишних нулей специально отсекается. В этом случае целые числа (например, индексы массивов) индицируются в естественной форме. Так, в нашем примере исполнение операторов

```
PRINT "A (" 2 ") = " ; A (2)
```

даст распечатку в виде

```
A (2) = 5
```

Для задания формата часто используются знаки #:

$\underbrace{\#\#\#}_{n_1}.\underbrace{\#\#\#\#}_{n_2}\uparrow$

Знак показательной формы ↑ при числах с фиксированной точкой опускается, например:

PRINT#.## X – печатается значение X в форме с фиксированной точкой при $n_1 = 1$ знаков до десятичной точки (запятой) и $n_2 = 2$ знаков после нее (например, если $X = 1{,}2345$, то распечатка даст 1.23);

PRINT#.##↑ X – печатается значение X в показательной форме при $n_1 = 1$ знаков до десятичной точки и $n_2 = 2$ знаков после нее (например, если $X = 12{,}345$, то распечатка дает 1.23E1).

Подобным образом задается формат и в расширенной версии языка Бейсик–Бейсик-80. Однако указание на вывод числа в показательной форме задается не одной, а четырьмя стрелками ↑↑↑↑. В Бейсик-80 предусмотрена выдача знака % перед числом, не помещающимся в заданный формат.

В некоторых версиях Бейсика используются и другие операторы для задания формы выдачи:

SET N – стандартной,

SET E n – показательной с нормализованной мантиссой и n ее цифрами после десятичной точки (n задается как число от 0 до 9),

SET F n – с фиксированной точкой и n знаками после десятичной точки,

SET $\emptyset$ – стандартной с усечением последней цифры мантиссы,

SET 5 – стандартной с округлением последней цифры мантиссы,

WAIT m – пауза в вычислениях с сохранением индикации на время $m\Delta t_n$, где Δt_n – длительность единичной паузы.

Приведенные операторы используются, в частности, в версии Бейсика ПЭВМ FX-702P. При этом $n = 1, 2, \ldots, 9$, $m = 0 - 999$ и $\Delta t = 0{,}02$ с. Пауза в вычислениях задается также с помощью оператора

```
[НС]  PAUSE ⟨Число m⟩
```

Если $m = 0$, то вычисления останавливаются до тех пор, пока не будет нажата любая клавиша пульта. Оператор PAUSE есть в большинстве версий Бейсика.

Оператор TAB n (от слова tabulation – табуляция) используется для *табуляции* – задания чисел в виде таблиц, обеспечивает печать заданного символа (или группы символов), начиная с n-й позиции строки. Если все символы не помещаются в данной строке, часть их переносится на следующую

строку. При использовании оператора TAB n следует помнить, что строка делится на некоторое число позиций M (от 32 у простых ПЭВМ до 80–120 у сложных). У некоторых версий Бейсика вместо оператора TAB используется оператор CSR (ПЭВМ FX-702P).

Действие оператора TAB n иллюстрирует следующая программа:

```
5 REM ДЕЙСТВИЕ МОДИФИКАТОРА TAB
10 FOR X=1 TO 5
20 PRINT'X=';X TAB(15)'LOG(X)=';LOG(X)
30 NEXT X
X= 1.000000     LOG(X)=  .000000
X= 2.000000     LOG(X)= 6.931472E-01
X= 3.000000     LOG(X)= 1.098612
X= 4.000000     LOG(X)= 1.386294
X= 5.000000     LOG(X)= 1.609438
```

Обратите внимание: все значения $\log x$ печатаются с позиции $n = 15$. Для некоторых ПЭВМ задание $n > M$ недопустимо и приводит к останову вычислений с индикацией ошибки. У других ПЭВМ задание $n > M$ возможно и означает перенос индикации на строку $\text{НС} = \text{int}(n/M)$, где int – целая часть числа в круглых скобках. В этом случае дробная часть (n/M) – $\text{frac}(n/M)$ определяет позицию в строке M $\text{frac}(n/M)$. Например, если $M = 32$, а $n = 70$, то $\text{int}(70/32) = 2$, и символы будут выведены на вторую строку и начинаться с позиции $\text{frac}(n/M)M = 6$. Отметим, что в этих примерах предполагается нумерация строк с 0.

Оператор

PRINT AT y, x ; ⟨Выводимые данные⟩

обеспечивает вывод данных (числовых значений констант, переменных, арифметических выражений или заключенных в кавычки символов и фраз), начиная со строки с номером y и позиции в ней с номером x. Действие оператора AT, применяемого в расширенных версиях Бейсика, можно рассмотреть на примере следующей программы:

```
  10 REM ОПЕРАТОР ВЫВОДА PRINT
  20 PRINT "ВЫВОД ТЕКСТА"
  30 LET X=PI: PRINT "PI=";X
  40 PRINT 1;2;3;4;5;6;7;8;9
  50 PRINT "ОДИН" "ДВА","ТРИ","Ч
ЕТЫРЕ",1,2,3,4
  60 PRINT TAB 10;"СДВИГ ВПРАВО
НА 10 ПОЗИЦИЙ"
  70 PRINT AT 10,15;"■■■"
ВЫВОД ТЕКСТА
PI=3.1415927   ■■■
123456789
ОДИН            ДВА
ТРИ             ЧЕТЫРЕ
1               2
3               4
          СДВИГ ВПРАВО НА 10 ПОЗ
ИЦИЙ
```

Операторы TAB и AT обеспечивают построение простейших графиков, составленных из символов языка Бейсик (см. гл. 5). Обычно эти операторы входят в состав конструкций с оператором PRINT.

В некоторых версиях Бейсика (например, ПЭВМ PC фирмы IBM, США) используется оператор вывода WRITE аналогично оператору PRINT с той разницей, что разделяет элементы данных запятыми

2.3. ВВОД ЧИСЛОВЫХ ДАННЫХ (ОПЕРАТОРЫ INPUT, LET, DATA, READ, RESTORE, SWAP)

Для ввода числовых данных с пульта ПЭВМ служит оператор INPUT (ввод):

[НС] INPUT {['Комментарий' ;] ⟨Переменная⟩ . . .}

Как уже отмечалось, оператор INPUT без комментария обычно ведет к останову с выдачей знака (?) перед вводом числового значения каждой переменной списка переменных. Например, при выполнении строки

```
1Ø INPUT X, Y, Z
```

ПЭВМ трижды выдаст знак вопроса. После ввода трех чисел (ввод фиксируется нажатием клавиши перевода строки) их значения будут последовательно присвоены переменным X, Y и Z.

Если в состав оператора INPUT вводится комментарий (в апострофах или кавычках), то выдача знака вопроса прекращается и на индикацию (обычно экран дисплея) выводится комментарий. Например, при выполнении строк

```
1Ø INPUT 'ВВЕДИТЕ X=' ; X
2Ø INPUT 'ВВЕДИТЕ Y=' ; Y
```

вначале на экране дисплея появится комментарий

ВВЕДИТЕ X=

После ввода числового значения X эта фраза исчезает и появляется новый комментарий

ВВЕДИТЕ Y=

После ввода числового значения Y последняя фраза исчезает и идет выполнение последующих строк программы. При этом переменные X и Y становятся определенными, т.е. приобретают введенные с пульта числовые значения.

У некоторых ПЭВМ комментарии при исполнении оператора INPUT выводятся в отдельную служебную зону экрана дисплея (вверху или внизу). Бывает весьма полезно с помощью оператора PRINT подтвердить ввод в рабочей зоне экрана или вывести введенное числовое значение на печать. Например, при выполнении строки

```
1Ø INPUT 'ВВЕДИТЕ X=' ; X : PRINT 'X=' ; X
```

вначале в служебной зоне появляется комментарий

ВВЕДИТЕ X=

После ввода числового значения X (например, числа 1.234) он исчезает, но в рабочей зоне появляется подтверждающее сообщение

X=1.234

У некоторых ПЭВМ возможна такая конструкция предложения ввода:

```
1Ø PRINT "X="; : INPUT "ВВЕДИТЕ X="; X: PRINT X
```

В этом случае в рабочей зоне экрана появляется сообщение

X =

в служебной зоне выводится комментарий

ВВЕДИТЕ X =

После ввода числового значения X комментарий исчезает, а надпись в рабочей зоне дополняется значением введенного числа, например

X = 1.234

У отдельных ПЭВМ вывод комментариев по оператору INPUT вообще не предусмотрен. В этом случае ввод комментария обеспечивается с помощью оператора

```
1Ø PRINT "ВВЕДИТЕ X" : INPUT X
2Ø PRINT "X=" ; X
```

при пуске программы появляется указание

```
ВВЕДИТЕ X
?
```

После ввода числового значения переменной X (например, 1.234) вместо вопросительного знака появляется подтверждение

X = 1.234

Оператор присваивания LET (от слова letter – помечать буквами):

$$[\text{НС}]\quad \text{LET}\ \langle\text{Переменная}\rangle = \left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}$$

служит для присвоения заданной переменной числового значения константы, переменной или арифметического выражения, например:

LET A=5 – переменной *A* присвоено значение константы 5,

LET A=B – переменной *A* присвоено значение переменной *B*,

LET Y=A*B/C – переменной *Y* присвоено значение арифметического выражения *AB/C*, при этом переменные *A*, *B* и *C* должны быть предварительно определены.

У многих ПЭВМ при наборе программы оператор LET можно не вводить: он автоматически появляется в листинге перед знаком равенства =. Наконец, некоторые ПЭВМ (например, FX-702P) вообще не имеют этого оператора. Присвоение переменным числового значения у них производится с помощью знака равенства, если он указан после переменной:

$$[\text{НС}]\quad \langle\text{Переменная}\rangle = \left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}$$

Оператор LET опускается и у одной из версий Бейсика (TDM-3M) систем подготовки программ на базе микроЭВМ "Электроника Д3-28".

С помощью операторов INPUT и LET можно задавать числовые значения

подряд нескольким переменным. Однако удобнее это делать с помощью специальных операторов.

Оператор DATA (данные) в виде

[НС] DATA {Число, число, . . . , число}

служит для ввода в ОЗУ ПЭВМ последовательности чисел, разделенных запятыми (списка данных). При этом в ОЗУ образуется стек, т.е. система замкнутых в кольцо регистров памяти, организованных так, что числа вводятся строго по порядку и выводятся в том же порядке (слева направо).

Для присвоения числовых значений переменным из списка данных оператором DATA служит оператор READ (чтение или считывание):

[НС] READ {Переменная, переменная, . . . , переменная}

При исполнении оператора READ первой переменной из его списка присваивается значение первого числа из списка оператора DATA, второй переменной – значение второго числа и т.д. Например, если задано DATA 1, 2, 3 и READ A, B, C, получим A = 1, B = 2 и C = 3.

В ходе исполнения оператора READ происходит поворот стека (подобно вращению барабана у револьвера). Поэтому после вывода последнего числа к выводу готово первое число. Однако с помощью специального оператора RESTORE (поворот) стек можно повернуть и установить в исходное состояние. Таким образом, имеется возможность задания числовых значений из списка данных оператора DATA различным переменным не только в строгом, но и в измененном порядке.

Приведенная ниже программа иллюстрирует совместное действие операторов DATA, READ и RESTORE:

```
10 REM ВВОД ДАННЫХ ОПЕРАТОРОМ DATA
20 DATA 1,2,3,4,5
30 REM ПРИСВОЕНИЕ ЧИСЛЕННЫХ ЗНАЧЕНИЙ ПЕРЕМЕННЫМ A,B,C,D,E
40 READ A,B,C,D,E
50 PRINT'A='A;'B='B;'D='D:PRINT'E='E;'D='D
60 REM ВОЗВРАТ СТЕКА ДАННЫХ
70 RESTORE
80 REM ПРИСВОЕНИЕ ЧИСЛЕННЫХ ЗНАЧЕНИЙ ПЕРЕМЕННЫМ F,L,M
90 READ F,L,M
100 PRINT'F='F;'L='L;'M='M
A= 1.000000000      B= 2.000000000      D= 4.000000000
E= 5.000000000      D= 4.000000000
F= 1.000000000      L= 2.000000000      M= 3.000000000
```

В некоторых версиях Бейсика предусмотрен оператор SWAP:

[НС] SWAP ⟨Переменная 1⟩, ⟨Переменная 2⟩

обеспечивает обмен значениями между переменными 1 и 2. Например, при выполнении программы

```
1Ø LET X=123: LET Y=456
2Ø SWAP X, Y
3Ø PRINT "X="; X, "Y="; Y
```

результат будет таким:

X = 456 Y = 123

т.е. значения переменных поменялись против первоначально заданных (в строке 10) значений X = 123 и Y = 456.

Все перечисленные выше операторы могут использоваться и для ввода символьных выражений и присвоения значений символьным переменным (особенности ввода символьных переменных рассматриваются в гл. 4).

2.4. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ, РАБОТА В РЕЖИМЕ КАЛЬКУЛЯТОРА

Форма записи арифметических выражений на языке Бейсик весьма близка к математической записи выражений в строчку. Этой форме в последнее время отдают все большее предпочтение: она облегчает набор простых и умеренно сложных формул. В связи с этим особых трудностей в составлении арифметических выражений на Бейсике не возникает. Напомним лишь, что деление обозначается знаком / (а не :), умножение знаком * (а не X или ·) и возведение в степень знаками ↑, ∧ или ¬ (зависит от типа ПЭВМ).

Арифметические операции могут выполняться с помощью операторов PRINT (см. выше) и конструкцией вида

LET ⟨Переменная⟩ = {Арифметическое выражение}
⟨Переменная⟩ = {Арифметическое выражение}

В арифметическом выражении могут использоваться константы, переменные и функции.

Выполнение основных арифметических операций с константами и переменными показано в приведенных ниже двух программах. Действие их столь очевидно, что не требует пояснений.

```
10 REM АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ С ЧИСЛАМИ
20 PRINT'3+2=';3+2
30 PRINT'9-12=';9-12
40 PRINT'3*4=';3*4
50 PRINT'3/4=';3/4
60 PRINT'3^4=';3^4
70 PRINT'5+3^4=';5+3^4
80 PRINT'(2+3)*(3+1)=';(2+3)*(3+1)
3+2=  5.000000000
9-12= -3.000000000
3*4=  1.200000000E 01
3/4=  7.500000000E-01
3^4=  8.100000000E 01
5+3^4=  8.600000000E 01
(2+3)*(3+1)=  2.000000000E 01


10 REM АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ
20 LET A=5:LET B=2
25 PRINT'A=';A,'B=';B
30 PRINT'A+B=';A+B
40 PRINT'A-B=';A-B
50 PRINT'A*B=';A*B
60 PRINT'A/B=';A/B
70 PRINT'A^B=';A^B
A=  5.000000000      B=  2.000000000
A+B=  7.000000000
A-B=  3.000000000
```

```
A*B=   1.0000000000E 01
A/B=   2.500000000
A^B=   2.5000000000E 01
```

Функции языка Бейсик будут описаны в дальнейшем. Пока лишь отметим, что многие из них имеют форму записи также весьма близкую к общепринятой. Только функции записываются прописными буквами, например вычисление синуса переменной x ($\sin x$) как SIN(X). У некоторых версий Бейсика аргумент функции не обязательно должен заключаться в скобки, т. е. можно писать SIN X.

Для включения в бейсик-программу арифметического выражения вида

$$u = U_m \sin(2\pi f t + \varphi) \qquad (2.1)$$

прежде всего необходимо переобозначить переменные с учетом ограничений на вид имен переменных, присущих заданной версии Бейсика. Для простых версий Бейсика приемлемы, например, следующие замены: напряжение $u \leftarrow$ U, максимальное напряжение (амплитуда) $U_m \leftarrow$ A, частота $f \leftarrow$ F, время $t \leftarrow$ T, фаза $\varphi \leftarrow$ Q. Если число π выводится в виде #PI, то выражение (2.1) приобретает весьма наглядный вид

```
LET = A * SIN (2*#PI*F*T+Q)
```

Обратите внимание: обычно пропускаемый при записи математических выражений знак умножения должен обязательно присутствовать в арифметическом выражении для бейсик-программы.

У более сложных версий Бейсика возможно обозначение переменных строчными латинскими буквами и сочетаниями их с прописными. Это позволяет приблизить форму арифметических выражений для бейсик-программ к более естественной. Например, (2.1) при этом запишется в еще более наглядном виде: ($u \leftarrow$ u, $U_m \leftarrow$ Um, $\pi \leftarrow$ PI, $f \leftarrow$ f, $t \leftarrow$ t, $\varphi \leftarrow$ q):

```
LET u = Um * SIN(2*PI*f*t+q)
```

Даже из этого простого примера видно, что прямое копирование бейсик-программ для разных ПЭВМ невозможно. У одних версий Бейсика число π обозначается как #PI, у других как PI, у третьих как π, у четвертых его вывод вообще не предусмотрен. Различны возможности для наименования переменных. Это затрудняет в какой-то мере перевод программ и использование на ПЭВМ другого типа, чем тот, для которого были разработаны программы. Однако эти трудности не следует и преувеличивать. Указанные различия достаточно просты и формальны. Пользователь, хорошо освоивший свою ПЭВМ, нередко способен учесть их прямо по ходу ввода программы. Кроме того, в большинстве версий Бейсика предусмотрено обозначение переменных только прописной латинской буквой или буквой с последующей цифрой (от 0 до 9). Рекомендуется учитывать это уже при записи математических выражений, расчет по которым предполагается выполнять на ПЭВМ. Например, выражение

```
U = A · SIN(2 · PI · F · T + Q)
```

(2.2)

едва ли уступает по наглядности выражению (2.1), зато почти копирует выражение, допустимое для подавляющего большинства версий Бейсика (остается лишь заменить точку · на знак *).

При использовании арифметических выражений в бейсик-программах следует учитывать приоритет выполнения операций:

1 Выражение в скобках
2 Возведение в степень
3 Отрицание (знак –)
4 Умножение и деление
5 Целочисленное деление
6 Сложение и вычитание
7 Вычисление функций
8 Операции отношения ($A > B$ и др.)
9 Специальные логические операции в порядке их перечисления: НЕ, И, ИЛИ, исключающее ИЛИ, логическая импликация, логическая эквивалентность.

Учет приоритета операций важен даже в простейших случаях. Например, вычисление $3 + 5 \cdot 2$ даст 13, так как вначале будет выполнено умножение числа 5 на 2, а затем к результату прибавлено число 3. Если нужно выполнить операции в порядке следования их символов, т.е. сложить 3 и 5 и результат умножить на 2, то арифметическое выражение нужно записать в виде $(3 + 5) \cdot 2$. Скобки нарушают естественный приоритет операций.

Иногда возникают сомнения в выполнении одинаковых по приоритету операций, например $A \cdot B/C/D$. В таких случаях полезно помнить, что равноценные по приоритету операции выполняются слева направо. Итак, вначале вычисляется $A \cdot B$, затем полученное число делится на C и результат $(A \cdot B/C)$ еще раз делится на D.

В сомнительных случаях рекомендуется специально использовать скобки. Например, выражение ab/cd, вполне понятное при обычной математической форме записи, нельзя записать в виде $A*B/C*D$, так как это даст результат abd/c, а не ab/cd. Введя скобки, получим правильную запись $A*B/(C*D)$. Правомерна также и такая запись $(A \cdot B)/(C \cdot D)$ или $A \cdot B/C/D$. Число вводимых скобок обычно не ограничено.

Режим калькулятора, присущий ПЭВМ, позволяет проводить вычисления по арифметическим выражениям без ввода программ. Большинство операторов и функций языка Бейсик можно использовать в режиме калькулятора. В этом режиме вычисления выполняются набором инструкций на языке Бейсик с арифметическими выражениями, но без указания номеров строк. При этом для присвоения переменным числовых значений используются операторы LET, DATA и READ, а для вычисления арифметических выражений операторы PRINT и LET.

Пример. Вычислить значение U по выражению (2.2), если A=1Ø, F=3, T=1 и Q=0,2 (углы в радианах).

Способ 1. Применение оператора LET.

LET A=1Ø – переменной A присвоено значение 10,

LET F=5 – переменной F присвоено значение 5,

LET T=1 – переменной T присвоено значение 1,

LET Q=Ø – переменной Q присвоено значение 0,

LET U=A*SIN(2*PI*F*T+ Q) – переменной U присвоено численное значение вычисленного арифметического выражения,

PRINT U –получаем результат U=1.9866933 (значение переменной U).

Способ 2. Применение только оператора PRINT.

```
PRINT 10*SIN(2*PI*3*1+.2)
```

В способе 1 значения всех переменных после вычислений сохраняются в памяти ПЭВМ (это эквивалентно вычислениям на микрокалькуляторе с вводом исходных данных в регистры памяти [3]). Способ 2 проще, но в нем значения переменных в арифметическом выражении заданы в виде констант и после вычисления уже не сохраняются.

Все операции в режиме калькулятора завершаются нажатием клавиши перевода строки (ПС, ENTER, EXE, RETURN и т.д.).

Некоторые специализированные ПЭВМ (например, FX-702P) с функциями мощного микрокалькулятора имеют дополнительные удобства в режиме калькулятора, например не требуют набора оператора PRINT (см. способ 2), ряд функций (статистический, корреляционный и регрессионный анализ, вычисление гиперболических и обратных гиперболических функций, вычисление факториала, преобразование угловых мер и координат и т.д.) у них реализован микропрограммно.

Обычно режим калькулятора используется для однократных вычислений малой сложности. В отличие от обычных микрокалькуляторов [3], ПЭВМ и калькуляторы, программируемые на языке Бейсик, выполняют вычисления по предварительно подготовленной и отредактированной перед пуском формуле. Это резко снижает вероятность ошибок в процессе вычислений, особенно если расчетная формула сложна.

Режим калькулятора очень полезен в ходе редактирования и проверки программ (см. § 6.2), вводится автоматически при всякой остановке программы. Таким образом, пользователь получает возможность проверить вычисления, ранее проведенные по программе, в режиме калькулятора, сравнить их и сделать надлежащие выводы о правильности работы программы. В режиме калькулятора можно проводить также предварительную подготовку данных (например, вводить число 1/3 в таком именно виде или π^2 в виде PI ↑ 2), а также обработку данных в конце вычислений (например, умножение результата на константу, его преобразование и т.д.).

2.5. ВЫЧИСЛЕНИЕ ЭЛЕМЕНТАРНЫХ АЛГЕБРАИЧЕСКИХ ФУНКЦИЙ (EXP, EXT, LN, LOG, LGT, SOR)

К стандартным элементарным алгебраическим функциям языка Бейсик относятся следующие функции: вычисление экспоненты e^x натурального логарифма $\ln x$, десятичного логарифма $\log x$ и квадратного корня $\sqrt{x}$. Все эти функции вычисляются специальными численными методами, реализованными микропрограммно.

Например, экспоненциальная функция может быть вычислена по ее разложению в ряд

$$e^x = 1 + \frac{x}{1} + \frac{x^2}{1 \cdot 2} + \frac{x^3}{1 \cdot 2 \cdot 3} + \ldots + \frac{x^i}{1 \cdot 2 \cdot 3 \ldots}$$

Этот ряд легко вычисляется с помощью только арифметических операций. Однако погрешность вычислений довольно медленно уменьшается с ростом числа включенных в вычисления членов. Отброшенные члены создают методическую погрешность, которая должна быть достаточно мала.

У большинства ПЭВМ погрешность вычисления не должна превышать ±1 младшего разряда. При этом приходится использовать десятки членов разложения ряда. Разумеется, есть специальные приемы уменьшения числа вычислений. Тем не менее вычисления функций требуют большего числа арифметических операций и соответственно заметно большего времени вычислений, чем при проведении элементарных арифметических операций.

Функция EXP вида

$$[\text{HC}]\quad \text{EXP}\ (\left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\})$$

обеспечивает вычисление функции e^x, где x задано константой, числом или арифметическим выражением, а $e = 2{,}718281828459045$. Встречается также функция EXP вида

$$[\text{HC}]\quad \text{EXP}[\,(\,]\left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}[\,)]$$

т. е. если аргумент функции — константа или переменная, он не заключается в круглые скобки.

Для вычисления значения 10^x служит функция EXT вида

$$[\text{HC}]\quad \text{EXT}(\left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\})$$

В ряде версий Бейсика эта функция отсутствует.

Для вычисления натурального логарифма $\ln x$ обычно используется функция LOG вида

$$[\text{HC}]\quad \text{LOG}(\left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\})$$

Для вычисления десятичного логарифма $\log x$ служит функция LGT вида

$$[\text{HC}]\quad \text{LGT}(\left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\})$$

Иногда вместо функции LGT используется функция LOG 1∅.

У некоторых ПЭВМ (например, FX-702P) встречаются иные формы записи функций логарифмирования:

$$[\text{HC}]\quad \text{LN}[(\,]\left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ (\text{Выражение})\end{matrix}\right\}[\,)\,]$$ для функции $\ln x$,

[НС] LOG[(]{Константа / Переменная / (Выражение)}[)] для функции $\log x$.

Наличие скобок не обязательно, так можно писать LOG X или LN 5.

Для вычисления квадратного корня $\sqrt{x}$ служит функция SQR (от слова square – квадратный корень) вида

[НС] SQR({Константа / Переменная / Выражение})

Встречается и другая форма этой функции

[НС] SQR[(]{Константа / Переменная / (Выражение)}[)]

Программа, приведенная ниже, иллюстрирует вычисления элементарных алгебраических функций:

```
10 REM ВЫЧИСЛЕНИЕ ОСНОВНЫХ АЛГЕБРАИЧЕСКИХ ФУНКЦИЙ
20 LET X=2:PRINT'X=';X
30 PRINT 'EXP(X)=';EXP(X)
40 LET Y=LOG(X):PRINT'LOG(X)=';Y
50 PRINT'LGT(X)=';LGT(X)
60 PRINT'SQR(X)=';SQR(X)
X=  2.000000000
EXP(X)=  7.389056099
LOG(X)=  6.931471806E-01
LGT(X)=  3.010299957E-01
SQR(X)=  1.414213562
```

При использовании этих функций необходимо знать их особенности, хорошо описанные в математической литературе [33]. Так, следует помнить, что функция e^x вычисляется до тех пор, пока ее значения находятся в пределах разрядной сетки ПЭВМ. Так, если она сверху ограничена значением $9{,}99999999 \cdot 10^{99}$, то значения x должны быть меньше 231 (при $x = 230$ $e^x = 7{,}722 \cdot 10^{99}$). Логарифмические функции могут иметь любой аргумент $x > 0$ (в пределах разрядной сетки ПЭВМ). Отрицательные числа не имеют логарифмов. Функция SQR(X) справедлива для любых $x \geqslant 0$.

Если аргумент алгебраических функций или их значения выходят за допустимые пределы, ПЭВМ выводит на индикацию указание об ошибке. Для последующих вычислений нужно устранить причины, приводящие к этому.

У многих ПЭВМ функция $\log x$ отсутствует. В этом случае для вычисления $\log x$ можно использовать выражение $\log x = \ln x / \ln 10 = 0{,}4342945 \ln x$. Заметим, что лучше использовать не более короткое выражение $\ln x / \ln 10$, а последнее – с предварительным вычислением величины $1/\ln 10$. Это почти вдвое сокращает время вычисления функции $\log x$. Аналогично вычисляются логарифмы x при любом другом основании: $\log_a x = \ln x / \ln a$.

2.6. ВЫЧИСЛЕНИЕ ТРИГОНОМЕТРИЧЕСКИХ И ОБРАТНЫХ ТРИГОНОМЕТРИЧЕСКИХ ФУНКЦИЙ (SIN, COS, TAN, ASN, ACS, ATN, PI, RAD, DEG)

Тригонометрические функции занимают важное место в расчетах, выполняемых на ПЭВМ. Аргументом этих функций является угол φ, выраженный в радианах (рад). Тригонометрические функции можно определить, рассмотрев вращающийся на плоскости вектор длиной $r=\sqrt{x^2+y^2}$, где x и y – координаты его конца (рис. 2.3). Тригонометрические функции имеют период повторения 360°, или 2π рад.

Функция вычисления синуса аргумента φ ($\sin\varphi$) вида

[НС] SIN({ Константа / Переменная / Выражение })

вычисляет значение $\sin\varphi=y/r$.

Функция вычисления косинуса аргумента φ ($\cos\varphi$) вида

[НС] COS({ Константа / Переменная / Выражение })

вычисляет значение $\cos\varphi=x/r$.

Функция вычисления тангенса аргумента φ ($\mathrm{tg}\,\varphi$) вида

[НС] TAN({ Константа / Переменная / Выражение })

вычисляет значение $\mathrm{tg}\,\varphi=\sin\varphi/\cos\varphi=y/x$.

У некоторых ПЭВМ из этих трех функций вычисляется лишь одна $\mathrm{tg}\,\varphi$. Под аргументом φ подразумевается значение константы, переменной или арифметического выражения. Недостающие функции могут вычисляться с помощью соотношений $\cos\varphi=\sqrt{(1+\mathrm{tg}^2\varphi)^{-1}}$, $\sin\varphi=\sqrt{(1+1/\mathrm{tg}^2\varphi)}$.

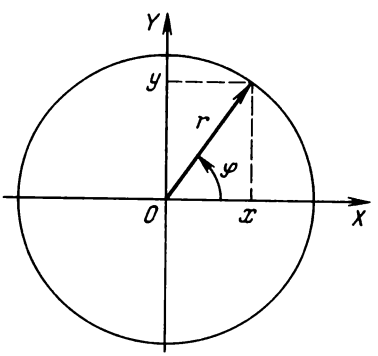


Рис. 2.3

Большинство современных ПЭВМ имеет все отмеченные тригонометрические функции, через них определяются также котангенс $\mathrm{ctg}\,\varphi=1/\mathrm{tg}\,\varphi$, секанс $\sec\varphi=1/\cos\varphi$, косеканс $\mathrm{cosec}\,\varphi=1/\sin\varphi$.

Функции $\sin\varphi$ и $\cos\varphi$ определены для любых φ, а $\mathrm{tg}\,\varphi$ для всех φ, кроме $\varphi=\pi/n+\pi/2$ (для последних $\mathrm{tg}\,\varphi\pm\infty$). Однако у многих ПЭВМ область определения тригонометрических функций ограничена. Например, у ПЭВМ FX-702P нельзя задавать φ выходящим за пределы $\pm 8\pi$ (в этом случае вычисления останавливаются с индикацией ошибки – выход аргумента за допустимые пределы). Поэтому

нужно следить за пределами изменения аргумента и программным путем ограничивать их. Учитывая периодичность тригонометрических функций, это несложно выполнить.

В общем случае углы тригонометрических функций задаются в радианах, градусах (2π рад соответствует 360°), иногда в градах (в прямом угле $\pi/2$ рад и и 100 град). Для перевода φ из радиан в градусы и наоборот используются следующие соотношения: $\varphi^\circ = \varphi_{\text{рад}} \cdot 180/\pi$, $\varphi_{\text{рад}} = \varphi^\circ \pi/180$. У ряда ПЭВМ для этого в наборе функций имеются функции RAD и DEG вида

[НС] RAD (Константа | Переменная | Выражение)
[НС] DEG (Константа | Переменная | Выражение)

Функция RAD преобразует аргумент, заданный в градусах, в аргумент, заданный в радианах. Функция DEG обеспечивает обратное преобразование. Перевод углов иногда выполняется и директивой MODE *N*, где *N* задает режим (моду) работы.

Как уже отмечалось, у большинства ПЭВМ предусмотрена выдача числа $\pi = 3{,}141592654$. Эта функция, обозначаемая #PI, PI или π, обеспечивает при отсутствии функций RAD и DEG перевод углов с помощью приведенных выше соотношений.

По заданному значению тригонометрической функции может быть найден ее аргумент φ. Для этого используются *обратные тригонометрические функции*. Расширенные версии Бейсика позволяют вычислять три обратные тригонометрические функции аргумента x: арксинус $\arcsin x$, арккосинус $\arccos x$ и арктангенс $\operatorname{arctg} x$, они задаются в виде

[НС] ASN (Константа | Переменная | Выражение)
[НС] ACS (Константа | Переменная | Выражение)
[НС] ATN (Константа | Переменная | Выражение)

У простых версий Бейсика из этих трех функций используется только одна $\operatorname{arctg} x$, остальные вычисляются по соотношениям

$\arcsin x = \operatorname{arctg}\,(x/\sqrt{1-x^2})$;
$\arccos x = \pi/2 - \operatorname{arctg}\,(x/\sqrt{1-x^2})$;
$\operatorname{arcctg} x = \pi/2 - \operatorname{arctg} x$.

Вычисление тригонометрических и обратных тригонометрических функций иллюстрирует следующая программа:

```
10 REM ВЫЧИСЛЕНИЕ ТРИГОНОМЕТРИЧЕСКИХ И ОБРАТНЫХ
20 REM ТРИГОНОМЕТРИЧЕСКИХ ФУНКЦИЙ
30 LET X=.5:PRINT 'X=';X;'РАД '
40 PRINT 'SIN(X)=';SIN(X)
50 PRINT 'COS(X)=';COS(X)
60 PRINT 'TAN(X)=';TAN(X)
70 PRINT 'ASN(X)=';ASN(X)
80 PRINT 'ACS(X)=';ACS(X)
90 PRINT 'ATN(X)=';ATN(X)
X=  5.000000000E-01 РАД.
SIN(X)=  4.794255386E-01
COS(X)=  8.775825619E-01
TAN(X)=  5.463024898E-01
ASN(X)=  5.235987756E-01
ACS(X)=  1.047197551
ATN(X)=  4.636476090E-01
```

У некоторых ПЭВМ аргумент всех перечисленных выше функций, заданный в виде константы или переменной, может не заключаться в скобки:

$$[\text{HC}]\ \text{ФУНКЦИЯ}\,[(]\begin{Bmatrix}\text{Константа}\\ \text{Переменная}\\ \text{(Выражение)}\end{Bmatrix}[)]$$

Справедливы, например, записи LET Y = SIN X, или LET Y = SIN 1.5 (переменной Y присваивается значение $\sin x$ или $\sin 1.5$), или LET Y = COS(2**PI*F*T).

2.7. ВЫЧИСЛЕНИЕ ГИПЕРБОЛИЧЕСКИХ И ОБРАТНЫХ ГИПЕРБОЛИЧЕСКИХ ФУНКЦИЙ (HSN, HCS, HTN, AHS, AHC, AHT)

Гиперболические и обратные гиперболические функции легко выражаются через экспоненциальные и логарифмические функции. Так, гиперболический синус $\operatorname{sh} x$, косинус $\operatorname{ch} x$ и тангенс $\operatorname{th} x$ связаны с функцией e^x соотношениями

$$\operatorname{sh} x = (e^x - e^{-x})/2;\ \operatorname{ch} x = (e^x + e^{-x})/2;$$

$$\operatorname{th} x = (e^x - e^{-x})/(e^x + e^{-x}) = \operatorname{sh} x/\operatorname{ch} x.$$

Ограничения на значения x такие же, как у функции e^x.

Для обратных гиперболических функций справедливы соотношения

$$\operatorname{arsh} x = \ln(x + \sqrt{1 + x^2}),\ -\infty < x < \infty;$$

$$\operatorname{arch} x = \ln(x + \sqrt{x^2 - 1}),\ x > 1;$$

$$\operatorname{arth} x = \ln\sqrt{(x + 1)/(1 - x)},\ -1 < x < 1.$$

Вычисление этих функций в простых версиях Бейсика специально не предусмотрено: они вычисляются по приведенным выше соотношениям.

Однако гиперболические и обратные гиперболические функции очень широко используются в практических расчетах, особенно в радиотехнических. Поэтому у большинства ПЭВМ, предназначенных для инженерных и научных расчетов (даже карманных, таких, как FX-702P), предусмотрено микропрограммное вычисление этих функций, задаваемых в виде

$$[\text{HC}]\ \langle\text{ФУНКЦИЯ}\rangle\ (\begin{Bmatrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражения}\end{Bmatrix})$$

или

$$[\text{HC}]\ \langle\text{ФУНКЦИЯ}\rangle\ [(]\begin{Bmatrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{Bmatrix}[)]$$

Здесь под обозначением ФУНКЦИЯ нужно понимать следующие функции, записанные прописными латинскими буквами: HSN – $\operatorname{sh} x$, HCS – $\operatorname{ch} x$, HTN – $\operatorname{th} x$, AHS – $\operatorname{arsh} x$, AHC – $\operatorname{arch} x$, AHT – $\operatorname{arth} x$.

Напоминаем, что аргумент x может быть константой, переменной или арифметическим выражением, его значения не должны превышать допусти-

мые пределы. Программа, показывающая выполнение вычислений этих функций, дана ниже, особых пояснений она не требует:

```
5 REM ВЫЧИСЛЕНИЕ ГИПЕРБОЛИЧЕСКИХ И ОБРАТНЫХ
10 REM ГИПЕРБОЛИЧЕСКИХ ФУНКЦИЙ
20 LETX=0.5:PRINT'X='X
30 PRINT'HSN(X)='HSN(X)
40 PRINT'HCS(X)='HCS(X)
50 PRINT'HTN(X)='HTN(X)
60 PRINT'AHS(X)='AHS(X)
70 PRINT'AHC(1/X)='AHC(1/X)
80 PRINT'AHT(X)='AHT(X)
X= 5.000000000E-01
HSN(X)= 5.210953055E-01
HCS(X)= 1.127625965
HTN(X)= 4.621171573E-01
AHS(X)= 4.812118251E-01
AHC(1/X)= 1.316957897
AHT(X)= 5.493061443E-01
```

2.8. ОСНОВНЫЕ ЧИСЛОВЫЕ И ДОПОЛНИТЕЛЬНЫЕ ФУНКЦИИ (ABS, SGN, INT, FRAC, PRC, RPC, DMS, RND, HEX$, OCT$)

Числовые функции расширяют возможности ПЭВМ в проведении различных расчетов. Основными числовыми функциями, входящими в любую версию Бейсика, являются следующие:

ABS (от слова absolute – абсолютное значение) – выделение абсолютного значения аргумента x (т.е. вычисление $|x|$ – модуля x);

SGN (от слова signum – знак) – определение знака числа (если функция дает +1, знак аргумента положителен, если 0, аргумент равен 0, и если –1, знак аргумента отрицателен);

INT (от слов integer portion – целая часть) – выделение целой части дробного аргумента;

CINT – функция, выделяющая целую часть аргумента с округлением числа (например, число 17,95 преобразуется в 18).

Общая форма записи этих функций подобна приведенной выше для гиперболических функций. Использование функций ABS, SGN и INT поясняет следующая программа:

```
10 REM ВЫЧИСЛЕНИЕ ОСНОВНЫХ ЧИСЛОВЫХ ФУНКЦИЙ
20 LET X=2:PRINT'X=';X
30 PRINT'ABS(-X)=';ABS(-X)
40 PRINT'INT(#PI)=';INT(#PI)
50 PRINT'SGN(X)=';SGN(X)
60 PRINT'SGN(0)=';SGN(0)
70 PRINT'SGN(-X)=';SGN(-X)
X=  2.000000000
ABS(-X)=  2.000000000
INT(#PI)=  3.000000000
SGN(X)=  1.000000000
SGN(0)=  .000000000
SGN(-X)= -1.000000000
```

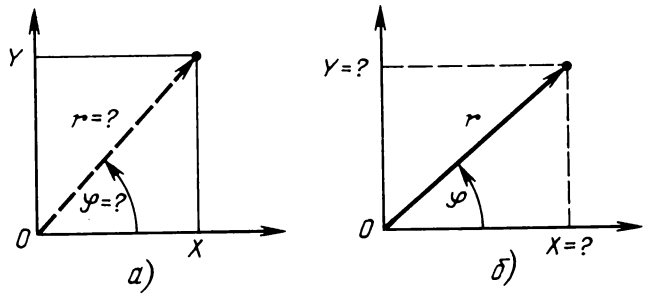


Рис. 2.4

В расширенных версиях языка Бейсик встречается также функция FPAC – выделение дробной части константы, значения переменной или арифметического выражения. Например, при выполнении строки

```
5Ø  LET A=FPAC(4/3)
```

переменная A получит значение 0,33333333 – дробная часть числа 4/3 = = 1,33333333.

У ПЭВМ, ориентированных на научные и инженерные расчеты (например, FX-702P), встречается ряд *дополнительных функций*:

Функция ! – вычисление факториала. Знак этой функции ставится после числа–константы, переменной или арифметического выражения в скобках. Например, 3!, A! при A=3 и (6/2)! дают значение факториала трех, равное 6 (напомним, что $N!=1$ при $N=0$ и $N!=1\cdot 2\cdot 3\ldots N$ при целых $N>>0$).

Функция RPC X, Y служит для преобразования декартовых координат X и Y точки на плоскости (рис. 2.4,*а*) в полярные координаты. При этом переменной X присваивается значение, равное длине вектора r, а переменной Y – значение угла φ. Функция PRC X, Y служит для преобразования параметров r =X и φ=Y в координаты X, Y точки (x, y) в декартовой системе координат (рис. 2.4,*б*).

Функция DEG (у ПЭВМ FX-702P она отличается от рассмотренной в § 2.6) служит для перевода угла, заданного в виде ⟨градусы, минуты, секунды и их доли⟩, в угол, заданный в градусах с десятичными долями. Например, если $\varphi=14^\circ 25'36''$, то выполнение команды Q=DEG (14.25.36) дает значение переменной $Q=14{,}42666667^\circ$.

Функция DMS (от слов degrees – градусы, minutes – минуты и seconds – секунды) преобразует угол, заданный в градусах с десятичными долями, в угол, заданный в градусах, минутах и секундах с десятичными долями. Например, если $A=14{,}2536^\circ$, то выполнение команды DMS A ведет к выдаче результата в виде $14^\circ 15' 12.96''$. Эта же функция обеспечивает преобразование времени – часы с десятичными долями преобразуются в часы, минуты и секунды с десятичными долями.

Функция RND (от слова round – округлять) служит для округления значения аргумента с выводом всех разрядов числа до 10^y. Эта функция записывается в виде

[НС] RND ({ Константа / Переменная / Выражение } , y)

Например, выражение RND(2*π*5, – 2) вычисляет длину окружности радиуса r = 5 см с выводом разрядов результата до 10^{-2} ($y = -2$). Результат вычислений будет 31,4 см (без округления 31,41592654 см).

В применении описанных выше функций особых трудностей обычно не встречается. Исключением является лишь функция INT. У некоторых ПЭВМ выделение целой части происходит просто отбрасыванием дробной части. Тогда INT(17.8) и INT(–17.8) дают соответственно результат 17 и –17. Однако обычно функция INT создает наибольшее целое число, меньшее или равное значению аргумента. Но тогда INT(17.8) =17, а INT(–17.8) =–18. В таком виде функцию INT можно использовать для округления до ближайшего целого числа, добавляя к аргументу 0,5. Например, INT(17.8 + .5) =18 и INT(–17,8 + .5) =–18.

Некоторые типы ПЭВМ способны на языке Бейсик обрабатывать числа в шестнадцатеричном и восьмеричном представлении. Для этого применяются функции:

HEX$ –преобразование числа из десятичной формы в шестнадцатеричную (например, HEX$ (32) дает число 20, которое является шестнадцатеричным представлением десятичного числа 32); обычно десятичные числа – аргумент функции HEX$ должны лежать в пределах от –32 768 до 65 535.

OCT$ –преобразование числа из десятичной формы в восьмеричную, например, OCT$ (24) дает число 30 – восьмеричное представление десятичного числа 24.

2.9. БЕЗУСЛОВНЫЕ И УСЛОВНЫЕ ПЕРЕХОДЫ (ОПЕРАТОРЫ GO, TO, ON–GOTO, IF, THEN, AND, OR, ELSE, UNLESS)

Большинство программ разветвляющиеся. Для обеспечения ветвлений служат безусловные и условные переходы.

Безусловный переход обеспечивается оператором GO TO (слова go to означают "идти к"). У ряда ПЭВМ эти слова пишутся слитно, как GOTO. Используются следующие виды таких переходов:

[НС] GO TO НС

[НС] GO TO { НС / Переменная / Выражение }

В первом случае переход происходит к строке с номером НС, заданным в виде константы. Например, команда GO TO 1ØØ или GOTO 1ØØ обеспечивает переход к строке с номером НС = 100. Во втором случае номер строки может задаваться не только константой, но и значением переменной или арифметического выражения. Этот случай соответствует возможности безусловных переходов с *косвенной адресацией*, когда адрес перехода (номер строки) указан значением переменной или арифметического выражения.

Косвенная адресация заметно расширяет возможности программирования, хотя на практике применяется довольно редко.

У некоторых ПЭВМ предусмотрена возможность ввода, хранения и исполнения нескольких самостоятельных программ – программных блоков. Так, у ПЭВМ FX-702P таких блоков 10, они обозначены как P∅, P1,..., P9 или в общем виде P*N*.

Оператор

$$[\text{НС}]\quad \text{GOTO\#}\left\{\begin{matrix} N \\ \text{Переменная} \\ \text{Выражение} \end{matrix}\right\}$$

обеспечивает вход в программный блок P*N* либо с прямой адресацией (GOTO *N*, где $N = 0, 1, 2, \ldots, 9$), либо с косвенной. Адрес перехода усекается до целого числа.

Если необходимо обеспечить последовательный безусловный переход по нескольким адресам, совместно с оператором GOTO используется *оператор-переключатель* ON. Конструкция переходов в этом случае имеет вид

$$[\text{НС}]\quad \text{ON}\left\{\begin{matrix} \text{Константа} \\ \text{Переменная} \\ \text{Выражение} \end{matrix}\right\}\ \text{GOTO}\ \{\text{НС1}, \text{НС2}, \text{НС3}, \ldots\}$$

где НС1, НС2, НС3, ... – последовательно возрастающие номера строк, к которым происходит переход. Обычно константа, переменная или выражение в фигурных скобках должны принимать значения 1, 2, 3, ..., что вызывает безусловный переход соответственно к строкам с номерами НС1, НС2, НС3, ... Например, выполнение строки

```
1∅ ON I GOTO 5∅, 1∅∅, 2∅∅
```

при I = 1 обеспечит переход к строке 50, при I = 2 к строке 100 и при I = 3 к строке 200. Однако встречаются версии Бейсика, у которых переход к строке НСN происходит, если значение константы, переменной или выражения в фигурных скобках равно НСN. Так, в приведенном примере, если I = 100, переход произойдет к строке 100.

Условные переходы простейшего типа имеют конструкцию вида

$$[\text{НС}]\quad \text{IF}\left\{\begin{matrix} \text{Константа} \\ \text{Переменная} \\ \text{Выражение} \end{matrix}\right\}\ \begin{matrix} \text{Знаки} \\ \text{отношения} \\ \text{(условия)} \end{matrix}\ \left\{\begin{matrix} \text{Константа} \\ \text{Переменная} \\ \text{Выражение} \end{matrix}\right\}\ \begin{matrix} \text{THEN} \\ \text{GOTO} \end{matrix}\ \text{НС}$$

При этих переходах происходит сравнение числовых значений констант, переменных или арифметических выражений, стоящих до и после знака отношения (слово if означает если, then – тогда). Если заданное отношение (условие) выполняется то происходит переход к строке с номером НС.

Знаки отношения:

= равно,	< меньше,
≠ или ⟨ ⟩ не равно,	≥ или >= больше и равно,
> больше,	≤ или <= меньше и равно.

Если условие перехода не выполнено, то происходит переход к выполнению следующего оператора или (у некоторых версий) следующей строки. Во избежание недоразумений с различной трактовкой условных переходов при выполнении условия перехода операторы условных переходов описанного типа рекомендуется ставить в конец строки либо давать их в отдельной строке.

Более сложная конструкция условного перехода

$$[\text{НС}]\ \left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}\ \begin{matrix}\text{Знаки}\\ \text{отношения}\\ \text{(условия)}\end{matrix}\ \left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}\ \begin{matrix}\text{THEN}\\ \text{GOTO}\end{matrix}\ \left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}$$

обеспечивает не только прямые, но и косвенные переходы при выполнении заданного условия. Например, выполнение следующих строк:

```
1Ø  IF X>Ø THEN 1ØØ  (Прямая адресация)
```

или

```
1Ø  IF X>Ø THEN 2*N  (Косвенная адресация)
```

обеспечивает прямой переход к строке 100 (первый вариант перехода) либо косвенный переход к строке 2N (второй вариант) при $X>0$. Если $X\leqslant 0$, будет выполняться следующая за предложением команда (строка).

У сложных версий Бейсика имеется ряд дополнительных конструкций условных переходов, например, такая

$$[\text{НС}]\ \text{IF}\left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}\ \begin{matrix}\text{Знаки}\\ \text{отношения}\\ \text{(условия)}\end{matrix}\ \left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}\ \text{THEN}\ \{\text{Инструкция}\}$$

В этом случае при выполнении условия выполняется предложение после слова (модификатора) THEN {Инструкции} (операторы, функции, директивы). Например, если выполняется строка

```
1Ø  IF X=Ø THEN PRINT "X=Ø": STOP
```

то при X=Ø будут выполнены команды PRINT "X=Ø" и STOP, т.е. будет выведено на индикацию сообщение X=Ø и произойдет останов вычислений (команда STOP). При невыполнении условия происходит переход на следующую строку.

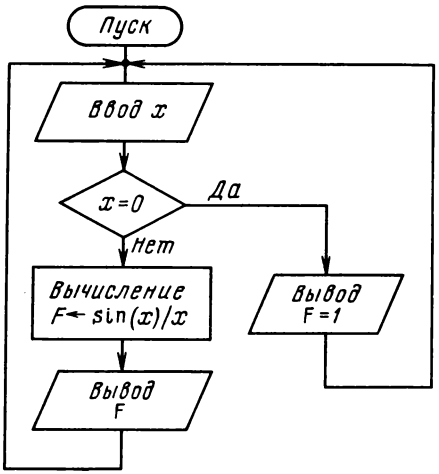
У некоторых ПЭВМ (например, FX-702P) вместо слова THEN в этой конструкции используется знак ; (THEN создает предшествующую конструкцию перехода). Возможна и подобная запись условного перехода:

```
1Ø  IF X=Ø THEN GOTO 5Ø
```

(если X=Ø, переход к строке 50, иначе к следующей строке).

В качестве примера построения разветвляющейся программы с условными и безусловными переходами рассмотрим вычисление функции

$$(\sin x)/x = \begin{cases}1, \text{ если } x = 0,\\ (\sin x)/x, \text{ если } x \neq 0.\end{cases}$$




```
10 INPUT "ВВЕДИТЕ X="X
20 GOSUB 40: PRINT "SIN(X)/X="S
30 GO TO 10
40 IF X=0 THEN LET S=1: RETURN
50 LET S=SIN(X)/X: RETURN
```

Рис. 2.5

Эту функцию нельзя вычислять как $(\sin x)/x$ при $x=0$: получается неопределенность 0/0. Деление на 0 ПЭВМ классифицирует как ошибку, хотя в данном случае неопределенность устраняется и $(\sin 0)/0=1$. Таким образом, алгоритм вычислений будет следующим:

1. Вводим значение x.
2. Сравниваем x с нулем.
3. Если $x \neq 0$, вычислим $\sin(x)/x$ и выводим результат на индикацию.
4. Если $x = 0$, выводим на индикацию тождественно известный результат – число 1.
5. Переходим к п. 1 – вводим новые значения x.

Схема этого алгоритма и соответствующая ему программа представлены на рис. 2.5.

У ПЭВМ, допускающих разделение программ на блоки, применяется также следующая форма условного перехода:

$$[\text{НС}]\left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}\ \begin{matrix}\text{Знаки}\\ \text{отношения}\\ \text{(условия)}\end{matrix}\ \left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}\ \text{THEN}\ [\#]\left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}$$

Знак # после THEN указывает, что при выполнении условия происходит условный переход с прямой или косвенной адресацией к программной области, адрес перехода усекается до целого числа. Отсутствие знака # указывает, что адресом является номер строки.

В наиболее сложных версиях языка Бейсик (например, ПЭВМ MSX, ZX-Spectrum) в состав условных переходов могут вводиться модификаторы AND (и), OR (или) и NOT (нет). Например, при выполнении строки

```
10 IF A=0 AND B=0 AND C=0 THEN PRINT "U=0"
```

печатается сообщение U=0 (переменные равны 0), если A, и B, и C равны 0. В противном случае выполняется следующая строка программы. Аналогично при выполнении строки

```
10 IF A=0 OR B=0 OR C≠ 0 THEN LET U=1
```

если A, или B, или C не равны 0, переменной U присваивается значение 1.

Наконец, встречается наиболее полная форма условных переходов:

[НС] { Список констант, переменных и выражений с операторами AND и OR } Знаки отношения (условия) THEN

{Инструкция} ELSE {Инструкция}

В этом случае при выполнении условия (или условий при вводе модификаторов AND и OR в условия) выполняются инструкции после слова THEN, в противном случае – инструкции после модификатора ELSE (иначе). Часть формы, заключенная в квадратные скобки, не является обязательной, например:

```
1ØØ  IF A AND B=Ø AND C=Ø THEN
PRINT "U=Ø" ELSE PRINT "U=1"
```

При выполнении этой строки, если A, и B, и C равны 0, печатается сообщение U = Ø, если иначе, – сообщение U = 1.

Из описанного следует, что применение модификаторов AND OR и ELSE позволяет строить конструкции программ, не относящиеся непосредственно к условным переходам. Они превращаются в условные переходы при применении оператора GOTO, например:

```
1Ø  IF  A = Ø AND B = Ø THE GOTO 15Ø
ELSE GOTO 3ØØ
```

– если A и B равны 0, переход к строке 150, если иначе, – к строке 300.

У некоторых версий Бейсика возможности модификаторов значительно расширены, в их число входит и слово IF (если). Например, может действовать такая конструкция:

```
1Ø  LET U=1 IF A=Ø AND B=Ø NO C=Ø
```

– переменной U присваивается значение 1, если A и B равны 0, но C не равно 0. В таких конструкциях применяется также модификатор UNLESS (если не). При его применении инструкции выполняются, если условие не соблюдается. Например, выполнение строки

```
5Ø  PRINT "X="; X UNLESS X=Ø
```

ведет к распечатке значения переменной X, если X не равно 0.

2.10. ОРГАНИЗАЦИЯ ЦИКЛОВ (ОПЕРАТОРЫ FOR, TO, STEP, NEXT, WHILE, UNTIL)

Многие вычисления желательно повторять циклически. Например, часто требуется вычислять значение какой-либо функции несколько раз (см. пример вычисления $\sin(x)/x$, приведенный выше). Этот пример показывает, что циклы можно организовать с помощью операторов условных и безусловных переходов. Однако построение программ с циклами, повторяющимися заданное число раз, упрощается, если использовать специальные операторы циклов: FOR (для), TO (до), STEP (шаг) и NEXT (следующий).

Структура цикла на языке Бейсик выглядит следующим образом:

[НС] FOR $\alpha=\beta$ TO γ [STEP δ]
[НС] ⟨Тело цикла⟩
[НС] NEXT α

Здесь α – управляющая переменная; β – константа, переменная или арифметическое выражение, которые задают начальное значение переменной α; γ – константа, переменная или арифметическое выражение, с численным значением которых сравнивается управляющая переменная α; δ – константа, переменная или арифметическое выражение, которые задают шаг изменения δ переменной α. При отсутствии необязательного элемента конструкции (в квадратных скобках) приращение δ автоматически задается равным 1.

Выполнение цикла происходит следующим образом:

1. Вначале однократно вычисляются значения β, γ и δ, запоминаются и потому в дальнейшем (в ходе выполнения цикла) меняться не могут. Задание $\delta=0$ недопустимо и воспринимается как ошибка.

2. Переменной цикла α задается начальное значение β.

3. Проверяется значение α на его допустимость. Значение считается допустимым, если при положительном шаге ($\delta>0$) $\alpha\leqslant\gamma$ или при отрицательном шаге ($\delta<0$) $\alpha\geqslant\gamma$.

4. Если значение α допустимо, выполняется тело цикла, иначе происходит переход к выполнению инструкции, следующей за NEXT α.

5. После завершения выполнения тела цикла производится приращение α на величину δ и выполняются операции, начиная с п. 3.

При выходе из цикла управляющая переменная обычно сохраняет свое последнее значение. Однако у некоторых ПЭВМ этого не происходит, и управляющая переменная становится неопределенной. Внутри цикла допустимо изменение управляющей переменной α (но не β, γ и δ). Допустим также выход из цикла с помощью операторов условных переходов. Однако вход в тело цикла извне, как правило, недопустим и воспринимается ПЭВМ как ошибка. Циклы в ряде версий Бейсика нельзя вводить в состав операторов условных переходов IF.

Как изменяется управляющая переменная цикла при задании положительного ($\delta>0$) и отрицательного ($\delta<0$) приращения, показывают приведенные ниже программы:

```
10 REM ЦИКЛ С ПОЛОЖИТЕЛЬНЫМ ПРИРАЩЕНИЕМ X
20 FOR X=-2 TO 2 STEP .5
30 PRINT 'X='X
40 NEXT X
X=-2.000000000
X=-1.500000000
X=-1.000000000
X=-5.000000000E-01
X=  .000000000
X= 5.000000000E-01
X= 1.000000000
X= 1.500000000
X= 2.000000000
```

```
10 REM ЦИКЛ С ОТРИЦАТЕЛЬНЫМ ПРИРАЩЕНИЕМ X
20 FOR X=2 TO -2 STEP -.5
30 PRINT 'X='X
40 NEXT X
X= 2.000000000
X= 1.500000000
X= 1.000000000
X= 5.000000000E-01
X=  .000000000
X=-5.000000000E-01
X=-1.000000000
X=-1.500000000
X=-2.000000000
```

Как отмечалось, исключение оператора STEP δ ведет к автоматическому заданию значения $\delta = 1$:

```
10 REM ЦИКЛ БЕЗ ОПЕРАТОРА STEP
20 FOR X=-2 TO 2
30 PRINT 'X='X
40 NEXT X
X=-2.000000000
X=-1.000000000
X=  .000000000
X= 1.000000000
X= 2.000000000
```

Это можно использовать для вычисления факториала $N! = 1\ 2\ 3 \ldots N$ при $N \geqslant 1$, включая особый случай $N! = 0! = 1$:

```
10 REM ВЫЧИСЛЕНИЕ ФАКТОРИАЛА N!
20 INPUT'ВВЕДИТЕ N='N
30 PRINT'N='N
40 LETR=1
50 FOR I=1 TO N
60 LETR=R*I:NEXT I
70 PRINT'N!='R
80 GOTO 20
N=  .000000000
N!= 1.000000000
N= 1.000000000
N!= 1.000000000
N= 1.000000000E 01
N!= 3.628800000E 06
```

Циклы широко применяются, когда необходимо осуществить табуляцию какой-либо функции. На рис. 2.6 дан пример программы табуляции экспоненциальной функции e^x и схема алгоритма этой циклической программы:

```
10 REM ТАБУЛЯЦИЯ ФУНКЦИИ EXP(X)
20 FOR X=20 TO -20 STEP -4
30 PRINT'X=';X;TAB(20);'EXP(X)=';EXP(X)
40 NEXT X
X=  2.000000000E 01  EXP(X)=  4.851651954E 08
X=  1.600000000E 01  EXP(X)=  8.886110520E 06
X=  1.200000000E 01  EXP(X)=  1.627547914E 05
X=  8.000000000      EXP(X)=  2.980957987E 03
X=  4.000000000      EXP(X)=  5.459815003E 01
```

```
X=   .000000000       EXP(X)=  1.000000000
X= -4.000000000       EXP(X)=  1.831563889E-02
X= -8.000000000       EXP(X)=  3.354626279E-04
X= -1.200000000E 01   EXP(X)=  6.144212353E-06
X= -1.600000000E 01   EXP(X)=  1.125351747E-07
X= -2.000000000E 01   EXP(X)=  2.061153622E-09
```

Возможно также применение циклов для присвоения переменным массива значений из списка оператора DATA с помощью оператора READ (см. также § 2.1):

```
10 REM ЗАДАНИЕ РАЗМЕРНОСТИ МАССИВА ИЗ N ЧИСЕЛ
20 LET N=5:DIM A(N)
30 REM ЗАДАНИЕ БЛОКА ДАННЫХ
40 DATA 1,2,3,4,5,6,7,8,9
50 REM ПРИСВОЕНИЕ ПЕРЕМЕННЫМ МАССИВА ЧИСЛЕННЫХ ЗНАЧЕНИЙ
60 FOR I=1 TO 5:READ A(I):NEXT I
70 REM РАСПЕЧАТКА ПЕРЕМЕННЫХ
80 FOR I=1 TO 5:PRINT!1.0!'A('I')='!F1.9!A(I):NEXT I
A( 1)= 1.000000000
A( 2)= 2.000000000
A( 3)= 3.000000000
A( 4)= 4.000000000
A( 5)= 5.000000000
```

Некоторая осторожность требуется при создании циклов с арифметическими выражениями для β, γ и δ. Из-за присущих ПЭВМ погрешностей иногда (обычно в самый неподходящий момент) цикл выполняется не так, как задумано, если конечное значение α окажется не точно равным γ. Например, цикл

```
1Ø FOR I=Ø TO 2 ↑ 3
2Ø PRINT I: NEXT I
```

может повторяться не $2^3 = 8$ раз, а лишь 7 раз, так как ПЭВМ вычисляет 2^3 как 7,99999999. В таких случаях можно создать искусственные условия для правильного выполнения цикла, добавив к γ небольшое число, например:

```
1Ø FOR I=Ø TO 2 ↑ 3 + .1
2Ø PRINT I: NEXT I
```

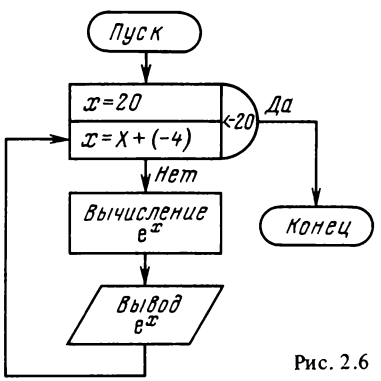


Рис. 2.6

Еще раз отметим, что описанная ситуация не является характерной. Однако подобный прием исключает ее в принципе. Возможность дополнительного изменения управляющей переменной в ходе выполнения операций тела цикла иллюстрирует следующая программа:

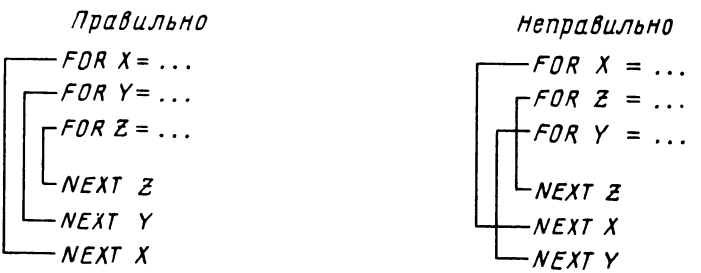
```
10 REM ЦИКЛ С ДОПОЛНИТЕЛЬНЫМ ИЗМЕНЕНИЕМ УПРАВЛЯЮЩЕЙ
20 REM ПЕРЕМЕННОЙ ВНУТРИ ТЕЛА ЦИКЛА
30 FOR I=0 TO 10: LET I=I+1
40 PRINT 'I='I:NEXT I
I= 1.000000000
I= 3.000000000
I= 5.000000000
I= 7.000000000
I= 9.000000000
I= 1.100000000E 01
```

Выход из тела цикла с помощью условного перехода иллюстрирует другая программа:

```
10 REM ВЫХОД ИЗ ЦИКЛА С ПОМОЩЬЮ
20 REM УСЛОВНОГО ПЕРЕХОДА В ТЕЛЕ ЦИКЛА
30 LET Y=5:FOR X=0 TO 10
40 IF X=Y THEN 70
50 PRINT'X='X
60 NEXT X:STOP
70 PRINT'ВЫХОД ИЗ ЦИКЛА ПРИ X=Y=5'
X=  .000000000
X= 1.000000000
X= 2.000000000
X= 3.000000000
X= 4.000000000
ВЫХОД ИЗ ЦИКЛА ПРИ X=Y=5
```

Из этих примеров видно также, что предложения с операторами FOR – TO – STEP и NEXT могут стоять в любом месте строки. Однако хотя бы ради наглядности программ рекомендуется помещать предложение с операторами FOR – TO – STEP в начало строки, открывающей цикл, а оператор NEXT в конец строки, закрывающей цикл. Еще нагляднее программы, в которых эти предложения стоят на отдельных строках.

Циклы могут вкладываться друг в друга. Примеры правильного и неправильного вложений:



При правильном вложении цикл, начатый первым, должен закрываться последним, последний цикл должен закрываться первым и т.д. Допустимо

заголовки нескольких циклов и завершающие их операторы NEXT указывать на одной строке:

НС1 FOR X = . . . : FOR Y = . . . : FOR Z = . . .
НС2 Тела циклов
НС3 NEXT Z: NEXT Y: NEXT X

Как уже отмечалось, циклы могут создаваться и с помощью операторов условных переходов, однако это не только несколько усложняет программу, но и увеличивает время счета:

```
10 REM ОРГАНИЗАЦИЯ ЦИКЛА С ЗАДАННЫМ ЧИСЛОМ ПОВТОРЕНИЙ
20 REM С ПОМОЩЬЮ ОПЕРАТОРА УСЛОВНОГО ПЕРЕХОДА
30 LET X=0:REM ЗАДАНИЕ НАЧАЛЬНОГО X
40 PRINT'X='X:REM ПЕЧАТЬ ЗНАЧЕНИЙ X
50 LET X=X+.2:REM ЗАДАНИЕ ПРИРАЩЕНИЯ X
60 IF X<=1 THEN 40:REM УСЛОВНЫЙ ПЕРЕХОД
70 REM КОНЕЦ ЦИКЛА
X=  .000000000
X= 2.000000000E-01
X= 4.000000000E-01
X= 6.000000000E-01
X= 8.000000000E-01
X= 1.000000000
```

Циклы, создаваемые с помощью операторов условных переходов, более универсальны, чем описанные выше. Так, такие циклы могут выполняться с переменным шагом:

```
10 REM ОРГАНИЗАЦИЯ ЦИКЛА С ПЕРЕМЕННЫМ ШАГОМ
20 REM С ПОМОЩЬЮ ОПЕРАТОРОВ УСЛОВНОГО ПЕРЕХОДА
30 LET X=0: REM ЗАДАНИЕ НАЧАЛЬНОГО X
40 PRINT'X='X: REM ПЕЧАТЬ ЗНАЧЕНИЙ X
50 LET X=X+.1+X/2: REM ЗАДАНИЕ МЕНЯЮЩЕГОСЯ ПРИРАЩЕНИЯ
60 IF X<=1 THEN 40: REM УСЛОВНЫЙ ПЕРЕХОД
70 REM КОНЕЦ ЦИКЛА
X=  .000000000
X= 1.000000000E-01
X= 2.500000000E-01
X= 4.750000000E-01
X= 8.125000000E-01
```

Нередко требуется организация циклов, число которых заведомо неизвестно. Например, вычисление обратного гиперболического тангенса

$$\operatorname{arth} x = x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \ldots$$

можно организовать по следующему алгоритму:

1. Задаем значение x и обнуляем переменную S.

2. Организуем цикл с управляющей переменной i, меняющейся от начального значения 3 с шагом 2 (т.е. $i = 3, 5, 7, \ldots$) до неопределенного (или заведомо очень большого значения). В теле цикла вычисляем $d = x^i / i$ и $S = S + d$, последняя операция означает суммирование членов ряда, начиная со второго.

3. Если d больше заданной погрешности ϵ (например, $\epsilon = 1 \cdot 10^{-6}$), то повторяем цикл, иначе переходим к п. 4.

4. Вычисляем $\operatorname{arth} x = x + S$, выводим полученное значение на печать и возвращаемся к п. 1.

Схема алгоритма вычисления $\operatorname{arth} x$ и реализующая его программа приведены на рис. 2.7.

```
10 REM ВЫЧИСЛЕНИЕ ФУНКЦИИ АНТ(X) ПО ЕЕ РАЗЛОЖЕНИЮ
20 REM В РЯД С ЗАДАННОЙ ПОГРЕШНОСТЬЮ УСЕЧЕНИЯ РЯДА
30 INPUT'ЗАДАЙТЕ ПОГРЕШНОСТЬ E='E
40 INPUT'ВВЕДИТЕ ЗНАЧЕНИЕ X='X
50 LET I=1:LET S=0
60 LET Z=S:LET S=S+X^I/I
70 LET I=I+2:IF S-Z>E THEN 60
80 PRINT'ЗНАЧЕНИЕ АНТ(X)='S:GOTO 30:END
```

У некоторых версий Бейсика (мини-ЭВМ СМ-4, ПЭВМ IBM PC) имеются модификаторы цикла WHILE (пока) и UNTIL (до) или WEND. Конструкция такого цикла следующая:

$$[\text{НС}]\quad \text{FOR } \alpha=\beta \text{ TO } \gamma\ [\text{STEP } \delta] \left\{ \begin{matrix} \text{WHILE} \\ \text{UNTIL} \end{matrix} \text{ Условие} \right\}$$

$$\ldots \text{Тело цикла} \ldots \text{NEXT } \alpha$$

В этом случае модификатор WHILE обеспечивает выполнение инструкций тела цикла до тех пор, пока стоящее после WHILE условие соблюдается, т. е. если оно истинно. Модификатор UNTIL делает то же, но до тех пор, пока это условие ложно. Условие имеет ту же форму, что и в операторах IF, т. е. может содержать два арифметических выражения и знак условия.

2.11. ОРГАНИЗАЦИЯ ПОДПРОГРАММ (ОПЕРАТОРЫ GOSUB, RETURN, RET, ON – GOSUB)

В процессе вычислений часто необходимо неоднократно выполнять идентичные фрагменты программ, например несколько раз вычислять одну и ту же функцию при разных значениях ее аргументов. В этом случае программы можно значительно сократить, обеспечив вычисление необходимого фрагмента по специальной программе – *подпрограмме*. К ней можно обращаться из основной программы с помощью операторов GOSUB или GO SUB (от слов go subroutine – идти к подпрограмме). Особенность подпрограммы заключается в том, что после ее выполнения происходит возврат в основную программу. Для этого в конце подпрограммы ставится специальный оператор RETURN (возврат), иногда используется RET.

Таким образом, структура основной программы с подпрограммой имеет вид

```
НС1   ...  Основная программа
НС2   ...   GO SUB   НС4 :  ...
НС3   ...  Основная программа  ... :
НС4   ...  Подпрограмма
НС5   RETURN
```

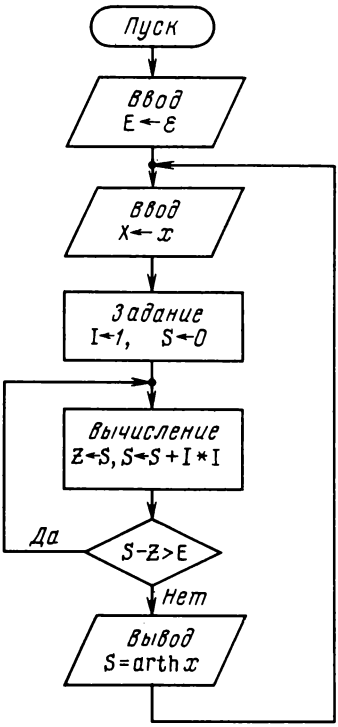


Рис. 2.7

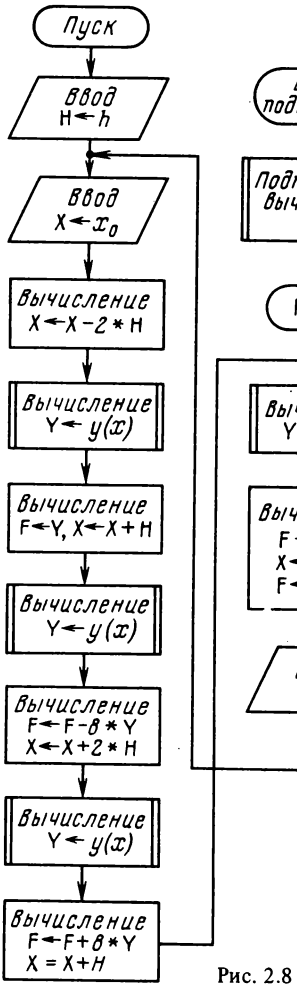


Рис. 2.8

Внутри одной подпрограммы может быть другая подпрограмма и т.д., т.е. подпрограммы могут быть вложены друг в друга. У большинства ПЭВМ число вложений не ограничено, но у некоторых ограничено до 10.

В общем случае обращение к подпрограмме имеет вид

$$[\text{НС}]\ \begin{matrix}\text{GO SUB}\\ \text{GOSUB}\end{matrix}\ [\#]\ \left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}$$

Если с оператором обращения используется константа НС (номер строки), то имеет место обращение к подпрограмме с прямой адресацией. Если же с оператором обращения используется переменная или арифметическое выражение, содержащее переменные, то обращение будет косвенное. Если

подпрограмма занимает отдельную программную область ПЭВМ (например, FX-702P), используется знак #. В этом случае программная область с подпрограммой должна завершаться оператором RETURN или RET.

Несмотря на свое название, подпрограммы обычно могут стоять не только под основной программой, но и в любом ее месте, в том числе и до основной программы. Возможно обращение к любой части подпрограммы. Подпрограмма может содержать несколько операторов возврата RETURN, но выполняется всегда тот, который встречается первым.

Вход в подпрограмму без обращения GOSUB, например с помощью безусловного или условного перехода воспринимается как ошибка и индицируется как ошибка вида "RETURN БЕЗ GOSUB". Недопустимо и обращение GOSUB к фрагменту программы, не содержащему оператора возврата RETURN. Такое обращение также воспринимается как ошибка вида "GOSUB БЕЗ RETURN".

Совместно с оператором GOSUB может использоваться оператор-переключатель

$$\text{ON}\left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}\text{GOSUB}\left\{\text{НС1, НС2, НС3, . . .}\right\}$$

Он действует так же, как с оператором ON – GOTO (см. § 2.9), с той разницей, что происходит не безусловный переход, а переход к выполнению подпрограмм, начинающихся со строк НС1, НС2, НС3 и т.д.

Для иллюстрации применения подпрограммы рассмотрим вычисление производной произвольной аналитически заданной функции $y(x)$ (оформим ее вычисление подпрограммой). Алгоритм решения этой задачи следующий:

1. Задаем шаг изменения h абсцисс x.
2. Задаем x_0 – исходное значение x.
3. Задавая $x = x_0 - 2h$, вычисляем по подпрограмме $y(x)$, получаем значение y_{-2}.
4. Задавая $x = x_0 - h$, вычисляем по подпрограмме $y(x)$, получаем значение y_{-1}.
5. Задавая $x = x_0 + h$, вычисляем по подпрограмме $y(x)$, получаем значение y_{+1}.
6. Задавая $x = x_0 + 2h$, вычисляем по подпрограмме $y(x)$, получаем y_{+2}.
7. Находим $dy/dx = (-y_{+2} + 8y_{+1} - 8y_{-1} + y_{-2})/(12h)$, выводим полученное значение на печать и переходим к п. 2.

Как видно из этого алгоритма, вычисление dy/dx потребовало вычисления $y(x)$ четыре раза при различных значениях аргумента x. Поэтому эти вычисления целесообразно организовать с помощью подпрограммы. Один из вариантов схемы этого алгоритма (с использованием значений y_{-2}, y_{-1}, y_{+1} и y_{+2} по мере их вычислений) и реализующая его программа представлены на рис. 2.8.

```
10 REM ЧИСЛЕННОЕ ДИФФЕРЕНЦИРОВАНИЕ ПРОИЗВОЛЬНОЙ АНАЛИТИЧЕСКОЙ
20 REM ФУНКЦИИ-ВЫЧИСЛЯЕТСЯ ПОДПРОГРАММОЙ СО СТРОКИ 100
```

```
30 INPUT'ЗАДАЙТЕ ШАГ H='H
40 INPUT'ЗАДАЙТЕ X0='X
50 LETX=X-2*H:GOSUB 100:LET F=Y
60 LETX=X+H:GOSUB 100:LETF=F-8*Y
70 LETX=X+2*H:GOSUB 100:LETF=F+8*Y
80 LETX=X+H:GOSUB 100:LETF=F-Y:LETX=X-2*H
90 LETF=F/12/H:PRINT'DY/DX='F:GOTO 40
100 REM ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ФУНКЦИИ Y(X)
110 LETY=EXP(-X*X/2)/SQR(2*PI)
120 RETURN:REM ВОЗВРАТ ИЗ ПОДПРОГРАММЫ
```

Со строки 100 в этой программе содержится подпрограмма вычисления функции начального распределения $y(x) = e^{-x^2/2}/\sqrt{2\pi}$. При $h = H = 0{,}001$ и $x_0 = 0{,}5$ вычисления по этой программе дают следующий результат: $dy/dx = -1{,}7603266 \cdot 10^{-1}$. Интересно отметить, что все цифры результата верны.

У ПЭВМ, имеющих программные блоки, подпрограммы можно записывать в них, не обращаясь к основной программе в ходе ввода подпрограмм. Например, если в блоке P0 находится основная программа, то в блоки P1–P9 можно ввести 9 подпрограмм. Дополнив приведенную выше программу (без подпрограммы) вводом числа $N = 1, 2, \ldots, 9$ и заменив обращения вида GOSUB 1ØØ на GOSUB #N, можно, задавая N, выполнять вычисления по любой из 9 заданных зависимостей $y(x)$.

Аналогичную возможность обеспечивает обращение к подпрограммам с косвенной адресацией. Так, в приведенном выше примере, задав обращение в виде GOSUB $N*100$ и организовав подпрограммы, начиная со строк $100(N=1)$, $200(N=2)$, $300(N=3)$ и т.д., можно заданием N указывать зависимость $y(x)$, производная которой вычисляется.

2.12. ОРГАНИЗАЦИЯ ФУНКЦИЙ ПОЛЬЗОВАТЕЛЯ (DEF FN, INKEY, KEY)

Нетрудно заметить, что обращение к подпрограммам менее удобно, чем к функциям. Действительно, надо указать номер строки или программного блока. Если в ходе модификации программы номер строки подпрограммы меняется, нужно найти и изменить все номера обращения к ней.

В связи с этим в языке Бейсик предусмотрены специальные средства для организации так называемых функций пользователя, т.е. функций, которые пользователь задает по своему усмотрению. Обращение к таким функциям производится столь же просто, как к "встроенным" функциям.

Функции пользователя могут быть двух типов: функции-выражения и функции-процедуры. Функции-выражения задают функции, определяемые как арифметические выражения, записываются в виде

$$[\text{HC}]\quad \text{DEF FN}\,\alpha(\langle\text{Аргументы}\rangle) = \left\{\begin{matrix}\text{Константа}\\ \text{Переменная}\\ \text{Выражение}\end{matrix}\right\}$$

Здесь FN α – имя функции пользователя, содержащее обязательно FN и имя α любой переменной.

Обращения к предварительно определенной функции-выражению задаются в виде

$$\text{FN}\,\alpha\left(\left\{\begin{array}{l}\text{Константы}\\ \text{Переменные}\\ \text{Выражения}\end{array}\right\}\right).$$

Здесь в фигурных скобках стоят аргументы функции-выражения. Число аргументов зависит от версии Бейсика. В простых версиях допускается один аргумент, в сложных – любое число. При обращении к функции FN α вначале вычисляется аргумент, а затем сама функция в операторе DEF FN α.

Применение оператора DEF FN α и функции FN α для вычисления гиперболического синуса HSN(X) по формуле $\text{sh}\,(x) = (e^{x} - e^{-x})/2$ иллюстрирует приведенная ниже программа

```
10 REM ЗАДАНИЕ ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ
20 DEF FNS(X)=(EXP(X)-EXP(-X))/2
30 REM ВЫЧИСЛЕНИЕ ЗАДАННОЙ ФУНКЦИИ S(X)=HSN(X)
40 LET X=1:PRINT'X='X
50 PRINT'HSN(X)='FNS(X)
60 PRINT'2*HSN(X)='2*FNS(X)
X= 1.000000000
HSN(X)= 1.175201194
2*HSN(X)= 2.350402387
```

Таким образом, функция-выражение обеспечивает задание оператором DEF FN α функции, которая может быть представлена одной формулой. Более сложные зависимости (представляемые рядом формул), требующие применения операций условных переходов или циклов, с помощью функций-выражений вычислять нельзя, они могут вычисляться с помощью обычных подпрограмм. Однако в некоторых версиях Бейсика можно задать такие функции и с помощью оператора DEF FN α в виде

[НС] DEF FN α (⟨Аргументы⟩) [:]
[НС] ⟨Тело вычисления функции⟩
[НС] FNEND

Таким образом, процедура-функция задается аналогично подпрограмме, но имеет заголовок в виде оператора DEF FN α и оператора конца FNEND (эквивалентный оператору RETURN в обычных подпрограммах). Однако обращение к процедуре-функции, аналогичное обращению к функции-выражению, происходит указанием не номера строки, а наименования функции α с помощью функции FN α. Это удобно при создании библиотек функций.

Задание функции

$$S(x) = \begin{cases} (\sin x)/x & \text{при } x \neq 0, \\ 1 & \text{при } x = 0 \end{cases}$$

обеспечивает приведенная ниже программа

```
10 REM S=SIN(X)/X
20 DEF FN S(X)
40 IF X<>0 THEN LET S=SIN(X)/X
30 IF X=0 THEN LET S=1
40 FNEND
```

Нередко возникает необходимость заставить ПЭВМ выполнить заданную пользователем функцию при нажатии клавиши с определенным символом. Для этого служат функции KEY или INKEY (key – клавиша, inkey – ввод с клавиши). Эти операторы используются в виде

[НС] [LET] ⟨Имя символьной переменной⟩ = KEY
[НС] [LET] ⟨Имя символьной переменной⟩ = INKEY [$]

Таким образом, функции KEY и INKEY$ присваивают переменной символьное значение, соответствующее нажимаемой клавише.

Пусть требуется вычислить $S(x)$ – см. пример выше при нажатии клавиши S, на нажатие других клавиш ПЭВМ не должна реагировать. Это делается с помощью программы, использующей оператор KEY (в версии Бейсика ПЭВМ FX-702P операторы INPUT и PRINT обозначаются сокращенно как INP и PRT):

```
2Ø  INP X
3Ø  S$=KEY: IF S$=" " THE 3Ø
4Ø  IF S$="S" THEN 6Ø
5Ø  GOTO 3Ø
6Ø  IF X=Ø; PRT "S(X)=1": GOTO 2Ø
7Ø  PRT "S(X)="; SIN X/X: GOTO 2Ø
```

В строке 20 вводится значение x и присваивается переменной X. В строке 30 выражение S$=KEY означает, что символьной переменной S$ присваивается значение нажимаемой клавиши, т.е. если нажать клавишу A, то значение S$="A", если нажать клавишу S, то S$="S" и т.д. Выражение IF S$=" " THEN 3Ø (строка 40) обеспечивает возврат к началу строки 30, если не нажата какая-то клавиша. Если нажать любую клавишу, кроме клавиши S (условие S$="S" не выполняется в строке 40), и выполнение строки 50 ведет также к возврату к строке 30. Таким образом, если не нажата какая-либо клавиша или нажата любая клавиша, кроме клавиши S, программа зацикливается, и строки 60 и 70 не выполняются. Если нажать клавишу S, то символьной переменной S$ будет присвоено значение символа "S". В результате после исполнения строки 40 произойдет переход к строке 60, т.е. будет вычислено значение $S(x) = \sin(x)/x$. После этого происходит возврат к строке 20 – вводу нового значения x.

Использование функции INKEY аналогично функции KEY. Например, строка

```
1Ø  LET A$=INKEY$: IF A$="A" THEN PRINT "ЗАДАНО A"
```

при нажатии клавиши A выдает сообщение

ЗАДАНО A

При нажатии любой другой клавиши это сообщение не выдается. Та же ситуация возникает и при выполнении более короткой строки

```
1Ø  IF INKEY$="A" THEN PRINT "ЗАДАНО A"
```

У некоторых ПЭВМ применяются обе функции: KEY для управления специальными функциональными клавишами и INKEY для генерации символа нажимаемой клавиши.

2.13. ГЕНЕРАЦИЯ СЛУЧАЙНЫХ ЧИСЕЛ (ОПЕРАТОРЫ RND, RAN#, RANDOMIZE)

Поведение многих объектов моделируется с помощью случайных чисел. Для этого в ПЭВМ предусмотрена функция генерации случайных чисел с равномерным распределением в интервале [0,1] с помощью оператора RND(X). Аргумент X в функции RND(X) не используется и может быть любым числом. В ряде версий Бейсика эта функция записывается как RND (без скобок и указания X).

У некоторых версий языка Бейсик (например, ПЭВМ FX-702P) функция RND используется для округления чисел (см. выше). Поэтому для генерации случайных чисел применяется функция RAN#.

Обычно различают два способа генерации случайных чисел. Первый – при каждом пуске генерация последовательности случайных чисел начинается с произвольного числа. Таким образом, запуская программу несколько раз, будем получать одни и те же последовательности случайных чисел. В этом случае без оператора RANDOMIZE при каждом пуске программы задаются различные случайные числа, а применение оператора RANDOMIZE обеспечивает генерацию последовательностей случайных чисел повторяющимися начальными числами каждой последовательности.

```
  10 REM СЛУЧАЙНЫЕ ЧИСЛА НЕПОВТО
     РЯЮЩИЕСЯ
  20 FOR I=1 TO 5
  30 PRINT "I=";I,
  40 PRINT "RND=";RND: NEXT I
  50 FOR I=6 TO 10
  60 PRINT "I=";I,
  70 PRINT "RND=";RND: NEXT I
I=1             RND=0.78868103
I=2             RND=0.15130615
I=3             RND=0.34892273
I=4             RND=0.16993713
I=5             RND=0.74623108
I=6             RND=0.98762085
I=7             RND=0.57159424
I=8             RND=0.87005615
I=9             RND=0.25434875
I=10            RND=.07699585
```

```
  10 REM СЛУЧАЙНЫЕ ЧИСЛА ПОВТОРЯ
     ЮЩИЕСЯ
  20 RANDOMIZE 10: FOR I=1 TO 5
  30 PRINT "I=";I,
  40 PRINT "RND=";RND: NEXT I
  50 RANDOMIZE 10: FOR I=6 TO 10
  60 PRINT "I=";I,
  70 PRINT "RND=";RND: NEXT I
I=1             RND=.012573242
I=2             RND=0.94412231
I=3             RND=0.80923462
I=4             RND=0.69281006
I=5             RND=0.96110535
I=6             RND=.012573242
I=7             RND=0.94412231
I=8             RND=0.80923462
I=9             RND=0.69281006
I=10            RND=0.96110535
```

В первой программе оператор RANDOMIZE отсутствует и генерируются два цикла с различными случайными числами, а во второй перед каждым циклом стоит оператор RANDOMIZE. В результате случайные числа в обоих циклах повторяются.

Генерация последовательностей случайных чисел, повторяющихся при каждом запуске, обычно необходима при оценке правильности выполнения программ по контрольному примеру. Например, сколько бы раз пользователь не запускал последнюю программу, он получит одни и те же данные (они приведены в программе). Однако запуск первой программы всегда ведет к выдаче разных случайных чисел.

Глава 3

ПРОГРАММИРОВАНИЕ СПЕЦИАЛЬНЫХ РАСЧЕТНЫХ ОПЕРАЦИЙ НА ПЭВМ

3.1. ОСОБЕННОСТИ ПРОГРАММИРОВАНИЯ С ПРИМЕНЕНИЕМ МАТРИЧНЫХ ОПЕРАЦИЙ (ОПЕРАТОРЫ MAT READ, MAT INPUT, MAT PRINT, ZER, CON, IND, TRN, INV)

Любая ПЭВМ способна оперировать одномерными и двухмерными массивами действительных чисел, образующими векторы и матрицы. *Матрица* – двухмерный массив чисел вида

$$\mathbf{A}=\begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{vmatrix} \quad \text{или} \quad \mathbf{A}=\begin{vmatrix} a_{00} & a_{01} & a_{0,n-1} \\ a_{10} & a_{11} & a_{1,n-1} \\ \dots & \dots & \dots \\ a_{m-1,0} & a_{m-1,1} & a_{m-1,n-1} \end{vmatrix},$$

заданный n столбцами и m строками. Произведение $m \times n$ называется размером матрицы. Большая буква (в нашем случае **A**) есть имя матрицы. Нумерация строк и столбцов может начинаться с 0 или 1 (зависит от конкретной версии Бейсика).

Действия с матрицами сравнительно просты, поскольку базируются на элементарных арифметических операциях. Однако число их обычно весьма значительно, поэтому расчеты с матрицами довольно трудоемки. Они заметно упрощаются при использовании специальных матричных операторов, входящих в развитые версии языка Бейсик, например Бейсик-плюс, Бейсик-3А-TDM.

Ввод матриц. Вводу матрицы предшествует резервирование в ОЗУ места под ее элементы. Для этого используется оператор DIM:

[HC] DIM ⟨$\alpha(m_1, n_1)$ [, $\beta(m_2, n_2)$, . . .]⟩

где $\alpha(m_1, n_1)$, $\beta(m_2, n_2)$ – индексированные переменные двухмерных массивов. Здесь m_i – максимальный номер строки; n_i – максимальный номер столбца. Если нумерация их начинается с 0, то m_i и n_i на 1 меньше значений m и n размера каждой матрицы. Имена массивов $\alpha, \beta, \dots$ могут быть любыми, принятыми в Бейсике, но не допускается их совпадение с именами одномерных массивов. В приведенных ниже программах нумерация элементов матриц начинается с нуля.

Ввод элементов матриц в ОЗУ осуществляется с помощью оператора DATA:

[НС] DATA ⟨Список элементов построчно⟩

Список элементов может быть дан для нескольких матриц, задается построчно вначале для первой матрицы, затем для второй и т.д. Элементы в списке разделяются запятыми.

Присвоение числовых значений введенных элементов индексированным переменным с именами α, β и т.д. обеспечивается матричным оператором MAT READ:

[НС] MAT READ ⟨α [, β, . . .]⟩

Если указан ряд имен, то вначале построчно задаются элементы первой индексированной переменной α, затем второй β и т.д. в соответствии с объявленной оператором DIM размерностью массивов.

У некоторых версий Бейсика (например, Бейсик-плюс) допустимо не указывать размерность массивов с помощью оператора DIM. В этом случае по умолчанию резервируется память под одномерный массив из 10 элементов или двухмерный из 10×10 элементов, если индексация начинается с 1, и 11×11 = 121, если начинается с 0. Размерность массива указывается в скобках после имен α, β, . . . переменных. Например, для выполнения строки

```
1ØØ MAT READ A, B, C
```

требуется предварительное задание размерности массивов A, B и C оператором DIM. Для выполнения строки

```
1ØØ MAT READ A(5), B(5,1Ø), C(5,1Ø)
```

этого не требуется, поскольку размерность массивов указана в списке переменных оператора MAT READ.

Ввод числовых значений элементов матриц с пульта ПЭВМ и присвоение индексированным переменным α, β, . . . числовых значений обеспечивается оператором

[НС] MAT INPUT ⟨α [, β, . . .]⟩

Он также может иметь две формы. Если размерность α, β, . . . объявлена оператором DIM, ее не нужно указывать при записи списка переменных. Например, строка

```
1ØØ MAT INPUT A, B, C
```

обеспечивает ввод элементов матриц A, B, C, размерность которых объявлена заранее. Размерность индексированных переменных может быть объявлена и в самом списке оператора MAT INPUT, например:

```
1ØØ MAT INPUT A(5), B(5,1Ø), C(5,1Ø)
```

При исполнении оператора MAT INPUT печатается знак вопроса (?), и нужно ввести список элементов построчно, вначале первой матрицы, затем второй и т.д. Если список неполный, то оставшимся элементам придаются нулевые значения. При переполнении списка лишние элементы игнорируются, останова ПЭВМ и индикации ошибок при этом не происходит.

Вывод матриц. Для вывода числовых значений элементов матриц α, β, ... построчно используется оператор MAT PRINT:

[НС] MAT PRINT α [, β, ...]

И в этом случае размерность $\alpha, \beta, \ldots$ объявляется заранее или указывается в списке имен индексированных переменных. Это же правило действует и в отношении других матричных операторов (см. ниже).

Задание матрицы. Матрица вида

$$\mathbf{A} = \begin{vmatrix} 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & 0 \\ 0 & 0 & \ldots & 0 \end{vmatrix}$$

называется *нулевой,* задается оператором

[НС] MAT α = ZER,

где α — имя матрицы. Так, оператор MAT A = ZER задает матрицу **A** со всеми нулевыми элементами:

```
10 REM ФОРМИРОВАНИЕ НУЛЕВОЙ МАТРИЦЫ А
20 DIM A(2,2)
30 MAT A=ZER
40 REM ВЫВОД МАТРИЦЫ А ПОСТРОЧНО
50 MAT PRINT A:END
```

При выполнении этой программы получим результат в виде девяти нулей.

Матрица вида

$$\mathbf{B} = \begin{vmatrix} 1 & 1 & \ldots & 1 \\ 1 & 1 & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots \\ 1 & 1 & \ldots & 1 \end{vmatrix},$$

у которой все $b_{ij} = 1$, называется *матрицей с единичными элементами*, задается оператором

[НС] MAT α = CON

Например, оператор MAT A = CON задает матрицу **A** с $a_{ij} = 1$:

```
10 REM ФОРМИРОВАНИЕ МАТРИЦЫ А С ЕДИНИЧНЫМИ ЭЛЕМЕНТАМИ
20 DIM A(2,2)
30 MAT A=CON
40 REM ВЫВОД МАТРИЦЫ А ПОСТРОЧНО
50 MAT PRINT A:END
```

При выполнении этой программы получим результат в виде девяти единиц.

Единичной матрицей, или матрицей с единичными диагональными элементами, называют матрицу

$$\mathbf{I} = \begin{vmatrix} 1 & \ldots & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ 0 & 0 & 1 & 0 \\ \ldots & \ldots & \ldots & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix},$$

у которой элементы $I_{k,k} = 1$, а остальные равны 0. Такая матрица задается оператором

[HC] MAT α = IDN

Например, оператор MAT A = IDN формирует единичную матрицу **A**:

```
10 REM ФОРМИРОВАНИЕ МАТРИЦЫ С ЕДИНИЧНЫМИ ДИАГОНАЛЬНЫМИ ЭЛЕМЕНТАМИ
20 DIM A(2,2)
30 MAT A=IDN
40 REM ВЫВОД МАТРИЦЫ А ПОСТРОЧНО
50 MAT PRINT A:END
```

При выполнении этой программы выводятся числа: 0; 0; 1; 0; 1; 0; 0; 0; 1.

Транспонирование матрицы. Транспонированной называется матрица, например с именем **B**, все элементы которой $b_{ji} = a_{ij}$, где a_{ij} – элементы исходной матрицы **A**. Другими словами, *транспонированная* матрица получается из исходной, если столбцы последней записать как строки. Например, если

$$\mathbf{A} = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{vmatrix}, \text{ то } \mathbf{B} = \begin{vmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{vmatrix}$$

Транспонирование обеспечивается оператором

[HC] MAT β = TRN (α)

где α – имя исходной матрицы; β – имя транспонированной матрицы. Перед использованием этого оператора необходимо зарезервировать память под матрицы α и β с помощью оператора DIM и задать исходную матрицу α. Имена α и β не должны совпадать. Размер матрицы β в общем случае не совпадает с размером матрицы α. В приведенной ниже программе исходная матрица **A** (см. пример выше) транспонируется в матрицу **B**:

```
10 REM ТРАНСПОНИРОВАНИЕ МАТРИЦЫ А
20 DIM A(2,2),B(2,2)
30 DATA 1,2,3,4,5,6,7,8,9
40 MAT READ A:MAT B=TRN(A)
50 PRINT!F1.5!'МАССИВ ТРАНСПОРТИРОВАННОЙ МАТРИЦЫ ПОСТРОЧНО'
60 MAT PRINT B:END
```

которая выводится на печать в виде цифр 1; 4; 7; 2; 5; 8; 3; 6; 9.

Инвертирование (обращение) матрицы. Матрица **B** или $\mathbf{A}^{-1}$ ($\mathbf{B} = \mathbf{A}^{-1}$) называется *инвертированной* по отношению к матрице **A**, если $\mathbf{AA}^{-1} = \mathbf{AB} = \mathbf{I}$, где **I** – единичная матрица. Здесь в последних выражениях подразумевается умножение матриц (см. далее). Инвертирование квадратных матриц выполняется с помощью оператора

[HC] MAT α = INV (β, γ)

где α – имя исходной матрицы; β и γ – имена двух вспомогательных одномерных массивов. Инвертированная матрица помещается на место исходной, причем одновременно вычисляется определитель D и его значение при-

сваивается элементу $\beta(0)$. Инвертирование выполняется методом Гаусса–Жордана. Векторы β и γ должны иметь одинаковую размерность n.

Приведенная ниже программа обеспечивает инвертирование матрицы

$$\mathbf{A} = \begin{vmatrix} 4 & 8 & 0 \\ 8 & 8 & 8 \\ 2 & 0 & 1 \end{vmatrix}$$

при задании вспомогательных векторов **B** и **C**:

```
10 REM ОБРАЩЕНИЕ МАТРИЦЫ A
20 DIM A(2,2),B(2,2),C(2,2)
30 DATA 4,8,0,8,8,8,2,0,1
40 MAT READ A: MAT A=INV(B,C)
50 PRINT!F1.5!'ОПРЕДЕЛИТЕЛЬ МАТРИЦЫ A  D='B(0)
60 PRINT'МАССИВ ОБРАЩЕННОЙ МАТРИЦЫ ПОСТРОЧНО'
70 MAT PRINT A:END
```

Выданные по этой программе результаты имеют следующий вид:

```
ОПРЕДЕЛИТЕЛЬ МАТРИЦЫ A D=9.6ØØØØE –01
МАССИВ ОБРАЩЕННОЙ МАТРИЦЫ ПОСТРОЧНО
 8.33333E – Ø2   –8.33333E – Ø2    6.66667E – Ø1    8.33333E – 02
 4.16667E – Ø2   –3.33333E – Ø1   –1.66667E – Ø1    1.66667E – 01
–3.33333E – Ø1
ОСТАНОВ СТРОКА 6Ø
```

Таким образом, в данном случае матрица $\mathbf{A} \leftarrow \mathbf{A}^{-1}$ принимает вид

$$\mathbf{A} = \begin{vmatrix} 8{,}33333 \cdot 10^{-2} & -8{,}33333 \cdot 10^{-2} & 6{,}66667 \cdot 10^{-1} \\ 8{,}33333 \cdot 10^{-2} & 4{,}16667 \cdot 10^{-2} & -3{,}33333 \cdot 10^{-1} \\ -1{,}66667 \cdot 10^{-1} & 1{,}66667 \cdot 10^{-1} & -3{,}33333 \cdot 10^{-1} \end{vmatrix} .$$

Умножение матриц. Умножение матрицы α размера $m \times n$ и матрицы β размера $n \times l$ создает матрицу $\gamma = \alpha \cdot \beta$, элементы которой вычисляются по формуле

$$\gamma_{kj} = \sum_{i=1}^{n} (\alpha_{ki} \cdot \beta_{ij}), j = 1, 2, \ldots, l; \quad k = 1, 2, \ldots, m.$$

Полученная матрица γ имеет размер $m \times l$. Умножение матриц обеспечивается с помощью оператора

[HC] MAT $\gamma = \alpha * \beta$

где α и β – имена перемножаемых матриц; γ – имя полученной матрицы. Массивы α, β и γ не должны совпадать.

Ниже приведена программа, обеспечивающая умножение матриц

$$\mathbf{A} = \begin{vmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 1 \\ 2 & 3 & 4 & 5 & 6 \\ 7 & 8 & 9 & 1 & 2 \end{vmatrix} \text{ и } \mathbf{B} = \begin{vmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \\ 1 & 2 & 3 \end{vmatrix} .$$

```
10 REM'УМНОЖЕНИЕ МАТРИЦ'
20 DIM A(3,4),B(4,2),C(3,2)
30 DATA 1,2,3,4,5,6,7,8,9,1,2,3,4,5,6,7,8,9,1,2
40 MAT READ A
```

```
50 DATA 1,2,3,1,2,3,1,2,3,1,2,3,1,2,3
60 MAT READ B
70 MAT C=A*B
80 PRINT !F1.5!'ВЫВОД ПОСТРОЧНО ЭЛЕМЕНТОВ МАТРИЦЫ C'
90 MAT PRINT C:END
```

Результат выполнения:

$$\mathbf{C} = \mathbf{A} \cdot \mathbf{B} = \begin{vmatrix} 15 & 30 & 45 \\ 31 & 62 & 93 \\ 20 & 40 & 60 \\ 27 & 54 & 81 \end{vmatrix}.$$

Арифметические операции с элементами матриц. Все элементы матрицы α могут быть независимо друг от друга преобразованы с помощью некоторого арифметического выражения γ.

[НС] MAT $\beta = \alpha$⟨Знак операции⟩ (γ)

Здесь β – полученная в результате преобразования матрица; α – исходная матрица; γ – арифметическое выражение, константа или переменная; знак операции: сложение +, вычитание –, деление / и умножение *. Массивы β и α могут совпадать, в этом случае результат заносится на место исходной матрицы.

В приведенной ниже программе выполняются действия (α = A, β = B, $\gamma = 2^3 - 6$)

$$\mathbf{A} \cdot (2^3 - 6) = \begin{vmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{vmatrix} \cdot (2^3 - 6).$$

```
10 REM УМНОЖЕНИЕ МАТРИЦЫ НА АРИФМЕТИЧЕСКОЕ ВЫРАЖЕНИЕ
20 DIM A(2,2),B(2,2)
30 DATA 1,2,3,4,5,6,7,8,9
40 MAT READ A
50 MAT B=A*(2^3-6)
60 PRINT!F1.5!'ВЫВОД ПОСТРОЧНО ЭЛЕМЕНТОВ МАТРИЦЫ B'
70 MAT PRINT B:END
```

Результат получаем в виде матрицы

$$\mathbf{B} = \begin{vmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{vmatrix}.$$

Пусть α и β – исходные матрицы одинакового размера. Их элементы α_{ij} и β_{ij} могут попарно складываться, отниматься, делиться и умножаться, создавая матрицу γ того же размера. Для этого используется оператор:

[НС] MAT $\gamma = \alpha$⟨Знак операции⟩ β

Знак операции: сложение +, вычитание –, деление / и умножение *. Например, поэлементное сложение для матриц A и B выполняется оператором MAT C = A + B.

Матричный оператор присваивания. Присвоение элементам матрицы α значений элементов матрицы β выполняется оператором

[НС] MAT $\alpha = \beta$

Например, оператор MAT A=B присваивает элементам матрицы **A** значения элементов матрицы **B**. Двухмерные массивы α и β (A и B в примере) должны иметь одинаковую размерность.

Решение систем линейных уравнений. Применение матричных операторов резко упрощает решение задач линейной алгебры. При этом все операции выполняются над именами матриц так же, как с обычными переменными. Так, система линейных уравнений

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n} = b_1,$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n} = b_2,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n$$

в матричной форме имеет вид

$$\mathbf{AX} = \mathbf{B}$$

где **A** – матрица коэффициентов: **X** и **B** – векторы неизвестных и свободных членов. Отсюда решение – вектор $\mathbf{X} = (x_1, x_2, \ldots, x_n)$ неизвестных находится в виде

$$\mathbf{X} = \mathbf{B}/\mathbf{A} = \mathbf{A}^{-1}\mathbf{B}.$$

Данный способ позволяет находить решение ряда уравнений с одинаковой матрицей **A** и разными векторами свободных членов **B**. При этом требуется лишь один раз обращать матрицу **A**. Обычно подобный способ применяется редко, поскольку при обращении матрицы число операций в 3–4 раза больше, чем при решении системы линейных уравнений одним из вариантов метода Гаусса [1]. Однако если ПЭВМ имеет специальные матричные операции, то такой способ не только заметно сокращает длину программы, но и обеспечивает уменьшение времени вычислений (до 10 раз). Последнее связано с выполнением операций на машинном языке ПЭВМ. Ниже приведена программа, реализующая описанный метод:

```
10 PRINT'ОБРАЩЕНИЕ МАТРИЦЫ A, ВЫЧИСЛЕНИЕ ЕЕ ОПРЕДЕЛИТЕЛЯ'
20 PRINT'И РЕШЕНИЕ СИСТЕМЫ УРАВНЕНИЙ A*X=B ПРИ РАЗНЫХ B'
30 INPUT'ЗАДАЙТЕ ЧИСЛО УРАВНЕНИЙ N=';N: N=N-1
40 DIM A(N,N),B(N),C(N,N),D(N,N),X(N)
50 FOR I=0 TO N:FOR J=0 TO N:PRINT!2.0!'ВВЕДИТЕ A'I+1','J+1
60 INPUT A(I,J):NEXT J:NEXT I:MAT A=INV(C,D)
70 PRINT!F1.5!'ВЫВОД ЭЛЕМЕНТОВ ОБРАЩЕННОЙ МАТРИЦЫ'
80 MAT PRINT A:PRINT'ОПРЕДЕЛИТЕЛЬ МАТРИЦЫ A  D='C(0)
90 INPUT'БУДЕТ ЛИ РЕШАТЬСЯ СИСТЕМА УРАВНЕНИЙ? ДА 1,НЕТ 0  'K
100 IF K=0 THEN PRINT'ВЫЧИСЛЕНИЯ ОКОНЧЕНЫ':END
110 FOR I=0 TO N:PRINT!2.0!'ВВЕДИТЕ B'I+1
120 INPUT B(I):NEXT I:MAT X=A*B
130 PRINT!F1.5!'МАССИВ НЕИЗВЕСТНЫХ СИСТЕМЫ'
140 MAT PRINT X:PRINT'КОНЕЦ':GOTO 90:END
```

В этой программе в строках 10–60 организовано задание числа уравнений N, ввод коэффициентов a_{ij} и обращение матрицы **A** (последний оператор строки 60). В строках 70 и 80 задается вывод элементов обращенной матрицы и определителя. В строках 90 и 100 задано окончание счета, если по программе предполагаются лишь операции с матрицами. Если предполагается решение систем уравнений, в строках 110 и

120 задастся ввод вектора **B** и вычисление вектора **X**. Затем в строках 130 и 140 задается вывод значений вектора **X** и переход к строке 90 для ввода нового вектора **B**.

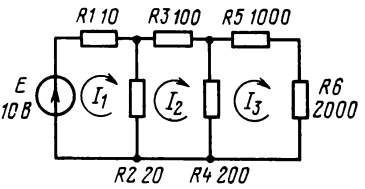


Рис. 3.1

П р и м е р. Для контроля приведенной выше программы вычислим токи $x_1 = I_1$, $x_2 = I_2$, $x_3 = I_3$ в линейной цепи (рис. 3.1). Воспользовавшись методом контурных токов, запишем систему уравнений

$$\begin{aligned} I_1R_1 + (I_1 - I_2)R_2 - E &= 0,\\ (I_2 - I_1)R_2 + I_2R_3 + (I_2 - I_3)R_4 &= 0,\\ (I_3 - I_2)R_4 + I_3(R_5 + R_6) &= 0. \end{aligned}$$

Подставив в нее сопротивления резисторов и напряжения E, приведя подобные члены, получим:

$$\begin{aligned} 30I_1 - 20I_2 + 0\cdot I_3 &= 10,\\ -20I_1 + 320I_2 - 200I_3 &= 0,\\ 0\cdot I_1 - 200I_2 + 3200I_3 &= 0. \end{aligned}$$

Решение этой системы дает значения токов: $x_1 = I_1 = 0{,}3484419$ А, $x_2 = I_2 = 0{,}0226629$ А и $x_3 = I_3 = 0{,}00141643$ А.

Решение системы линейных уравнений без матричных операторов. Систему линейных уравнений можно решить, не применяя матричные операторы. Поскольку у большинства версий Бейсика они отсутствуют, рассмотрим такое решение наиболее распространенным методом Гаусса. Он заключается в преобразовании коэффициентов (прямой ход) по формулам:

$$a_{ji} = -a_{ji}/a_{ii};\ a_{jk} = a_{jk} + a_{ji}a_{ik};\ b_j = b_j + a_{ji}b_i,$$

при $i = 1, 2, \ldots, n-1$; $j = i+1$; $i+2, \ldots, n$; $k = i+1$; $i+2, \ldots, n$. Эти преобразования приводят к вычислению $x_n = b_n/a_{nn}$. Далее выполняются преобразования (обратный ход):

$h = b_i$, $h = h - x_j a_{jj}$ для $i = n-1, n-2, \ldots, 2, 1$ и $j = i+1, i+2, \ldots, n$ и $x_i = h/a_{ii}$.

Эти преобразования дают неизвестные $x_{n-1}, x_{n-2}, \ldots, x_2, x_1$. Метод Гаусса реализует следующая программа:

```
05 PRINT'РЕШЕНИЕ СИСТЕМЫ ИЗ N ЛИНЕЙНЫХ УРАВНЕНИЙ'
10 INPUT 'ЧИСЛО УРАВНЕНИЙ N='N
20 DIM A(N,N),B(N),X(N)
30 FOR I=1 TO N: PRINT !2.0!'ВВОД КОЭФ. УРАВНЕНИЯ 'I
40 FOR J=1 TO N: INPUT A(I,J)
50 NEXT J: INPUT B(I):NEXT I
60 FOR I=1 TO N-1: FOR J=I+1 TO N
70 LETA(J,I)=-A(J,I)/A(I,I):FOR K=I+1 TO N
80 LETA(J,K)=A(J,K)+A(J,I)*A(I,K): NEXT K
90 LETB(J)=B(J)+A(J,I)*B(I): NEXT J: NEXT I
100 LETX(N)=B(N)/A(N,N)
110 FOR I=N-1 TO 1 STEP -1: LETH=B(I)
120 FOR J=I+1 TO N: LETH=H-X(J)*A(I,J):NEXT J
130 LETX(I)=H/A(I,I):NEXT I
140 PRINT 'КОРНИ СИСТЕМЫ УРАВНЕНИЙ'
```

```
150 FOR I=1 TO N: PRINT!2.0! 'X('I')=':PRINT!E! X(I)
160 NEXT I: END
```

Ее можно проверить по приведенному примеру.

Этот метод требует, чтобы коэффициенты a_{ii} не обращались в нуль, иначе наступит останов при вычислении $a_{ji} = -a_{ji}/a_{ii}$. В электрорадиотехнических расчетах равенство a_{ii} нулю обычно принципиально невозможно*. Однако существует множество разновидностей метода Гаусса, вообще некритичных к значениям a_{ii}, их описание можно найти в [29 – 31], а соответствующие программы на языке Бейсик в [1].

Полиномиальная аппроксимация и интерполяция. При расчетах и моделировании нелинейных устройств и систем часто возникает задача аналитического представления сложной нелинейной зависимости $f(x)$, заданной рядом узлов $f_i(x_i)$ с помощью некоторой достаточно простой аналитической функции $y(x)$. В качестве последней часто выбирают степенной полином

$$y(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0.$$

Если $f(x)$ задана $n+1$ узлом, то можно найти полином $y(x)$, значения которого $y(x)$ в точности равны $f(x)$ в узловых точках. Следовательно, имея $n+1$ значений абсцисс x_i и ординат y_i $(0 \leqslant i \leqslant n)$, коэффициенты $a_0, a_1, \ldots, a_n$ можно найти из решения системы линейных уравнений

$$\begin{aligned}
x_0^n a_n + x_0^{n-1} a_{n-1} + \ldots + x_0 a_1 + a_0 &= y_0,\\
x_1^n a_n + x_1^{n-1} a_{n-1} + \ldots + x_1 a_1 + a_0 &= y_1,\\
&\ldots\ldots\ldots\ldots\ldots\ldots\\
x_n^n a_n + x_n^{n-1} a_{n-1} + \ldots + x_n a_1 + a_0 &= y_n.
\end{aligned}$$

Этот подход (метод выбранных точек) реализуется следующей программой:

```
10 PRINT'ПОЛИНОМИАЛЬНАЯ АППРОКСИМАЦИЯ И ИНТЕРПОЛЯЦИЯ'
20 INPUT'ЗАДАЙТЕ ЧИСЛО ОТСЧЕТОВ N='N: M=N-1
30 DIM A(M,M),C(M,M),D(M,M),B(M),X(M)
40 FOR I=0 TO M:PRINT!2.0!'ВВЕДИТЕ X'I',Y'I
50 INPUT Z,B(I): A(I,0)=1: R=1:FOR J=1 TO M
60  R=R*Z: A(I,J)=R:NEXT J:NEXT I
70 MAT A=INV(C,D):MAT X=A*B
80 PRINT!F1.9!'КОЭФФИЦИЕНТЫ A0,A1,...,AM':MAT PRINT X
90 INPUT'ВВЕДИТЕ X='Z: Y=Z*X(M)
100 FOR I=M-1 TO 1 STEP-1: Y=(Y+X(I))*Z:NEXT I
110  Y=Y+X(0):PRINT'Y(X)='Y:GOTO 90:END
```

В строках 40, 50 и 60 организован ввод x_i и y_i, формирование коэффициентов $x_i^n, x_i^{n-1}, \ldots, x_i$ матрицы A при неизвестных $a_n, a_{n-1}, \ldots, a_1, a_0$ и ввод вектора B

---

* Для систем, составленных по методу узловых потенциалов.

ординат $y_0 - y_n$. Затем (строка 70) обращается матрица **A** коэффициентов и находится вектор $\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}$, содержащий искомые неизвестные $a_0, \ldots, a_n$.

Пример. Рассмотрим аппроксимацию N-образной вольт-амперной характеристики туннельного диода ЗИ202К (рис. 3.2), задав ее рядом точек:

i	0	1	2	3	4	5
$x = U$, В	0	0,2	0,4	0,6	0,8	1
$y = I$, мА	0	52	23	2	4	13

Введя эти данные ($N = 5$), получим: $a_0 = 0$, $a_1 = 746{,}333$, $a_2 = 3420{,}833$, $a_3 = 5734{,}375$, $a_4 = -4166{,}667$ и $a_5 = 1119{,}792$.

Интерполяция заключается в нахождении значений $f(x)$ в промежутках между узлами. Для этого в рассмотренной программе используются вычисления полинома $y(x)$ по схеме Горнера, когда полином представляется в виде

$$y(x) = (\ldots(a_n x + a_{n-1})\,x + a_{n-2})x + \ldots + a_1)x + a_0.$$

Эта схема заменяет медленные операции возведения в степень гораздо более быстрыми операциями умножения и сложения, реализована в строках 90–110 программы.

Пусть нужно найти ток I при $U = 0{,}05$ и 0,1 В. Задав $x = 0{,}05$, получим $y = I = 22{,}456$ мА (при $x = 0{,}1$ имеем $I = 45{,}454$ мА).

Решение систем нелинейных уравнений. Системы нелинейных уравнений вида:

$$F_0(x_0, x_1, x_2, \ldots, x_{n-1}) = 0,$$
$$F_1(x_0, x_1, x_2, \ldots, x_{n-1}) = 0,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$F_{n-1}(x_0, x_1, x_2, \ldots, x_{n-1}) = 0$$

или в матричной форме

$$\mathbf{F}_i\ (\mathbf{X}_i) = 0$$

чаще всего решаются методом Ньютона–Рафсона. При этом задается число уравнений $n = N$, относительная погрешность вычислений E и начальные значения переменных x_{00}, x_{01}, $x_{02}, \ldots, x_{0n-1}$, т. е. вектор $\mathbf{X}_{i0}$. Далее формируется матрица частных производных каждой функции (матрица Якоби)

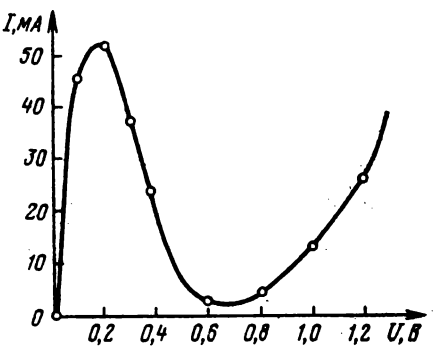


Рис. 3.2

$$\left|\frac{\partial \mathbf{F}_i}{\partial \mathbf{X}_i}\right| = \begin{vmatrix} \frac{\partial F_0}{\partial x_0} & \frac{\partial F_0}{\partial x_1} & \cdots & \frac{\partial F_0}{\partial x_{n-1}} \\ \frac{\partial F_1}{\partial x_0} & \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_{n-1}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial F_{n-1}}{\partial x_0} & \frac{\partial F_{n-1}}{\partial x_1} & \cdots & \frac{\partial F_{n-1}}{\partial x_{n-1}} \end{vmatrix}.$$

Поскольку аналитическое дифференцирование $(\frac{\partial F_i}{\partial x_i})$ не всегда возможно, целесообразно частные производные заменить конечно-разностными выражениями. В матричной форме это дает

$$\frac{\partial \mathbf{F}_i}{\partial \mathbf{X}_i} \approx \frac{\Delta \mathbf{F}_i}{\Delta \mathbf{X}_i} = \frac{\mathbf{F}_i(\mathbf{X}_i + \mathbf{H}_i) - \mathbf{F}_i(\mathbf{X}_i)}{\mathbf{H}_i},$$

где $\mathbf{H}_i$ – вектор малых приращений $\mathbf{X}_i$, например $\mathbf{H}_i = \mathbf{E}|\mathbf{X}_i|$.

Решение системы нелинейных уравнений методом последовательных приближений (итераций) записывается в виде рекуррентной формулы

$$\mathbf{X}_{i\,(m+1)} = \mathbf{X}_{i\,(m)} - (\frac{\partial \mathbf{F}_i}{\partial \mathbf{X}_i})^{-1} \mathbf{F}_i(\mathbf{X}_{i\,(m)}), \qquad (3.1)$$

где m – номер итерации (начиная с 0). Таким образом, новый вектор $\mathbf{X}_{i\,(m+1)}$ находится по старому $\mathbf{X}_{i\,(m)}$ с помощью операции инвертирования матрицы Якоби и соотношения (3.1). Итерации продолжают, пока

$$[\mathbf{X}_{i(m+1)} - \mathbf{X}_{i(m)}]/\mathbf{X}_{i(m+1)} > E, \qquad (3.2)$$

т.е. пока относительная погрешность каждого неизвестного x_i больше заданной. В противном случае счет останавливают и выводят на индикацию компоненты $x_0, x_1, x_2, \ldots, x_{n-1}$ вектора $\mathbf{X}_i$. Описанный метод реализует программа:

```
10 PRINT'РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ УРАВНЕНИЙ'
20 PRINT'МОДИФИЦИРОВАННЫМ МЕТОДОМ НЬЮТОНА.ВАР.2'
30 INPUT'ЗАДАЙТЕ ЧИСЛО УРАВНЕНИЙ N='N: N=N-1
40 DIM X(N),B(N),F(N),A(N,N),K(N,N),L(N,N)
50 INPUT'ЗАДАЙТЕ МАКСИМАЛЬНОЕ ЧИСЛО ИТЕРАЦИЙ M='M
60 INPUT'ЗАДАЙТЕ ОТНОСИТЕЛЬНУЮ ПОГРЕШНОСТЬ E='E: S=0
70 FOR I=0 TO N:PRINT!2.0!'ВВЕДИТЕ X'I'(0)':INPUT X(I):NEXT I
80 GOSUB 180 : MAT B=F
90 FOR J=0 TO N: Y=X(J): H=E*ABS(Y)
100  X(J)=Y+H:GOSUB 180:FOR I=0 TO N
110  A(I,J)=(F(I)-B(I))/H:NEXT I: X(J)=Y:NEXT J
120  S=S+1:IF S=M+1 THEN PRINT'ЧИСЛО ИТЕРАЦИЙ S=M':END
130 MAT A=INV(K,L):MAT B=A*F:MAT X=X-B
140  R=0:FOR I=0 TO N:IF ABS(B(I)/X(I))>E THEN  R=1
150 NEXT I:IF R=1 THEN 80
160 PRINT!F1.9!'МАССИВ НЕИЗВЕСТНЫХ СИСТЕМЫ'
170 MAT PRINT X:PRINT!2.0!'ЧИСЛО ИТЕРАЦИЙ S='S:END
180 REM'ПОДПРОГРАММА ВЫЧИСЛЕНИЯ F(I)=F(X(0),X(1),...,X(N-1))'
```

```
190  F(0)=X(0)+3*LGT(X(0))-X(1)*X(1)
200  F(1)=2*X(0)*X(0)-X(0)*X(1)-5*X(0)+1
210 RETURN:END
```

В строках 30–70 задается N, максимальное число итераций M, E и X_{i0}. В строке 80 осуществляется обращение к подпрограмме (со строки 180), обеспечивающей выполнение $F_i(m)$, после чего этому вектору приписывается имя B. Строки 100 и 110 формируют матрицу Якоби. В строке 120 организован счетчик числа итераций S и останов вычислений при $S=M$. Это необходимо для того, чтобы исключить зацикливание программы, если итерационный процесс не сходится (например, из-за неудачного выбора начальных значений компонентов вектора X_i). Обращение матрицы Якоби, вычисление $X_{i(m+1)}$ и проверка условия (3.2) производятся в строках 130–150. В строке 170 выводится на индикацию вектор неизвестных. Начиная со строки 180, пользователем задается подпрограмма вычисления функций $F_i(X_i)$. Она должна вычислять значения $F(I)=F_i(X_i)$ при $X(I)=x_i$ ($i=0, 1, \ldots, n-1$). В тексте программы записан контрольный пример для решения системы уравнений ($n=N=2$).

$$x_0 - 3\lg x_0 - x_1^2 = 0,$$
$$2x_0 - x_0 x_1 - 5x_0 + 1 = 0.$$

При $E=1\cdot 10^{-4}$, $M=10$ и $x_{00}=3{,}4$ и $x_{10}=2{,}2$ получим $S=3$ (т. е. результат получен за три итерации), $x_0=3{,}48744278$ и $x_1=2{,}261628631$.

Метод Ньютона–Рафсона широко применяется для анализа статического режима нелинейных электронных цепей и схем. Его сходимость зависит от вида нелинейных функций и выбора начальных условий.

3.2. ОПЕРАЦИИ, ФУНКЦИИ И СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ С КОМПЛЕКСНЫМИ ПЕРЕМЕННЫМИ

Наряду с действительными числами в расчетах, в частности электрорадиотехнических, широко используются комплексные числа. Эти числа можно представить в алгебраической форме $Z=a+ib=\mathrm{Re}Z+i\mathrm{Im}Z$, где $a=\mathrm{Re}Z$ – действительная часть числа; $b=\mathrm{Im}Z$ – мнимая часть числа. На плоскости (рис. 3.3) комплексное число характеризует точку с координатами a и b на действительной и мнимых осях.

Комплексные числа можно представить также в показательной форме. В этом случае их записывают как $Z=Me^{i\varphi}$, где $M=\sqrt{a^2+b^2}$ – модуль комплексного числа, а $\varphi=\mathrm{arctg}(b/a)$ – аргумент комплексного числа (см. рис. 3.3). Модуль комплексного числа характеризует длину вектора, проведенного из точки (0,0) к точке с координатами (a, b).

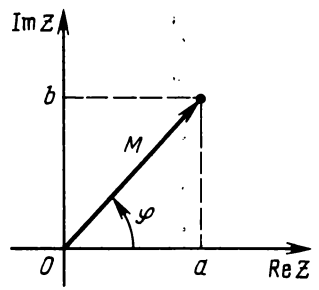


Рис. 3.3

Над комплексными числами могут проводиться арифметические операции, например сложение и вычитание:

$$Z_1 \pm Z_2 = (a_1 \pm a_2) + i(b_1 \pm b_2)$$

или умножение и деление

$$Z_1 Z_2 = (a_1 a_2)e^{i(\varphi_1+\varphi_2)} = (a_1 a_2 - b_1 b_2) + i(a_1 b_2 + b_1 a_2);$$

$$\frac{Z_1}{Z_2} = \frac{a_1}{a_2} e^{i(\varphi_1 - \varphi_2)} = \frac{a_1 a_2 + b_1 b_2}{a_2^2 + b_2^2} + i \frac{b_1 a_2 - a_1 b_2}{a_2^2 + b_2^2} .$$

При сложении и вычитании удобно использовать алгебраическую форму, при умножении и делении – показательную. Для перехода из показательной формы в алгебраическую используются соотношения

$a = M \cos \varphi$ и $b = M \sin \varphi$.

В результате арифметических операций над комплесными числами получается также комплексное число.

Все элементарные функции могут быть представлены с аргументом в виде комплексных чисел. Например, функция e^Z представляет собой число $e^Z = e^a \cos b + i(e^a \sin b) = a_1 + ib_1$, где $a_1 = e^a \cos b$ и $b_1 = e^a \sin b$ – действительная и мнимая части результата. Аналогично можно определить другие функции с комплексным аргументом.

Следовательно, арифметические операции с комплексными числами и вычисление элементарных функций комплексного переменного сводятся к простым вычислениям по формулам, которые можно найти в любой справочной литературе по математике, включая [1], где даны и отдельные бейсик-программы таких операций. Универсальная подпрограмма вычисления функций комплексного переменного приведена в § 7.1 данной книги.

Более сложной, но часто встречающейся в электрорадиотехнических расчетах задачей является решение системы линейных уравнений с комплексными коэффициентами

$$(a_{11} + ib_{11})\dot{x}_1 + (a_{12} + ib_{12})\dot{x}_2 + \ldots + (a_{1n} + ib_{1n})\dot{x}_n = c_1 + id_1,$$
$$(a_{21} + ib_{21})\dot{x}_1 + (a_{22} + ib_{22})\dot{x}_2 + \ldots + (a_{2n} + ib_{2n})\dot{x}_n = c_2 + id_2,$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$(a_{n1} + ib_{n1})\dot{x}_1 + (a_{n2} + ib_{n2})\dot{x}_2 + \ldots + (a_{nn} + ib_{nn})\dot{x}_n = c_n + id_n.$$

В матричной форме такую систему можно записать в простом виде

$$\dot{\mathbf{A}}\dot{\mathbf{X}} = \dot{\mathbf{B}},$$

где $\dot{\mathbf{A}}$ – матрица с комплексными коэффициентами $(a_{ij} + ib_{ij})$, $\dot{\mathbf{X}}$ – комплексный вектор неизвестных и $\dot{\mathbf{B}}$ – комплексный вектор свободных членов $(c_i + id_i)$.

Матрицу $\dot{\mathbf{A}}$ можно представить в виде обычной матрицы с действительными коэффициентами

$$\mathbf{A} = \begin{vmatrix} a_{11} b_{11} & a_{12} b_{12} & a_{1n} b_{1n} \\ a_{21} b_{21} & a_{22} b_{22} & a_{2n} b_{2n} \\ \ldots\ldots & \ldots\ldots & \ldots\ldots \\ a_{n1} b_{n1} & a_{n2} b_{n2} & a_{nn} b_{nn} \end{vmatrix} = |\operatorname{Re}\dot{\mathbf{A}} \operatorname{Im}\dot{\mathbf{A}}|,$$

в которую входят члены a_{ij} и b_{ij} в их естественном порядке. Эту матрицу можно записать и иначе:

$$\mathbf{A}=\left|\begin{array}{cccc} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \\ \hline b_{11} & b_{12} & \ldots & b_{1n} \\ b_{21} & b_{22} & \ldots & b_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ b_{n1} & b_{n2} & \ldots & b_{nn} \end{array}\right| = \left|\begin{array}{c} \mathrm{Re}\,\dot{\mathbf{A}} \\ \hline \mathrm{Im}\,\dot{\mathbf{A}} \end{array}\right| .$$

Для решения систем линейных уравнений с комплексными коэффициентами в матричном виде составляется расширенная матрица

$$\widetilde{\mathbf{A}}=\left|\begin{array}{cc} \mathrm{Re}\,\dot{\mathbf{A}} & \mathrm{Im}\,\dot{\mathbf{A}} \\ \hline \mathrm{Im}\,\dot{\mathbf{A}} & \mathrm{Re}\,\dot{\mathbf{A}} \end{array}\right| .$$

Эта матрица имеет размер $2n \times 2n$. Решение системы в этом случае имеет вид

$$\mathbf{X} = \widetilde{\mathbf{A}}^{-1}\mathbf{B},$$

где $\widetilde{\mathbf{A}}^{-1}$ – обращенная матрица $\widetilde{\mathbf{A}}$; $\mathbf{X} = \left|\begin{array}{c} \mathrm{Re}\,\dot{\mathbf{X}} \\ \hline \mathrm{Im}\,\dot{\mathbf{X}} \end{array}\right|$; $\mathbf{B} = \left|\begin{array}{c} \mathrm{Re}\,\dot{\mathbf{B}} \\ \hline \mathrm{Im}\,\dot{\mathbf{B}} \end{array}\right|$.

Этот метод реализован в следующей программе:

```
10 REM РЕШЕНИЕ СИСТЕМЫ ЛИНЕЙНЫХ УРАВНЕНИЙ
20 REM С КОМПЛЕКСНЫМИ КОЭФФИЦИЕНТАМИ
30 INPUT'ВВЕДИТЕ ЧИСЛО УРАВНЕНИЙ N=';N: M=N+N
40 DIM A(M,M),B(M),C(M),D(M),X(M)
50 FOR I=1 TO N: FOR J=1 TO N
60 PRINT!2.0!'ВВЕДИТЕ A 'I',';J: INPUT A(I,J)
70 PRINT!F1.9!A(I,J):  A(I+N,J+N)=A(I,J)
80 PRINT!2.0!'ВВЕДИТЕ B 'I',';J: INPUT A(I+N,J)
90 PRINT!F1.9!A(I+N,J):  A(I,J+N)=-A(I+N,J)
100 NEXT J: NEXT I
110 PRINT: PRINT'НАЙДЕТ ОБРАЩЕНИЕ МАТРИЦЫ'
120 MAT A=INV(C,D)
130 FOR I=1 TO N: PRINT!2.0!'ВВЕДИТЕ C'I
140 INPUT B(I): PRINT!F1.9!B(I)
150 PRINT!2.0!'ВВЕДИТЕ D'I: INPUT B(I+N)
160 PRINT!F1.9! B(I+N): NEXT I
170 MAT X=A*B: PRINT: PRINT'РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ'
180 FOR I=1 TO N:PRINT!2.0!'X'I' =';F1.9!X(I)'+J*('X(I+N)')'
190 NEXT I: GOTO 130: END
```

В строках 30–100 формируется матрица $\widetilde{\mathbf{A}}$, после чего она инвертируется (строки 110 и 120). В строках 130–150 задается вектор B, а в строках 170–190 вычисляется и выводится на печать вектор X.

Данный метод позволяет создавать довольно компактные программы. Однако он не вполне экономичен по емкости занимаемой памяти, поскольку матрица A имеет повышенный размер $(2n \times 2n)$. Поэтому этот метод целесообразен только при реализации с помощью матричных операторов, когда инвертирование матрицы $\widetilde{\mathbf{A}}$ выполняется достаточно быстро.

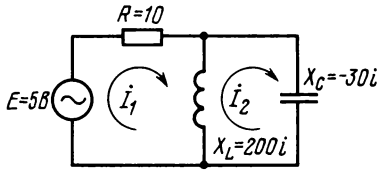


Рис. 3.4

Как отмечалось, большинство ПЭВМ имеет версии Бейсика без специальных матричных операторов. В этом случае решать систему линейных уравнений целесообразно иным методом – методом Гаусса (см. выше), но, заменив все операции над действительными числами операциями над комплексными числами. Построенная таким образом программа дана ниже:

```
10 PRINT 'РЕШЕНИЕ СИСТЕМЫ ИЗ N ЛИНЕЙНЫХ УРАВНЕНИЙ'
20 PRINT 'С КОМПЛЕКСНЫМИ КОЭФФИЦИЕНТАМИ'
30 INPUT'ЧИСЛО УРАВНЕНИЙ N='N:DIM A(N,N),B(N,N),C(N),D(N)
40 DIM K(N),L(N):FOR I=1 TO N:PRINT!2.0!'ВВОД КОЭФ. УРАВНЕНИЯ'I
50 FOR J=1 TO N:PRINT'ВВЕДИТЕ RE A('I','J')'
60 INPUT A(I,J):PRINT'ВВЕДИТЕ IM A('I','J')'
70 INPUT B(I,J):NEXT J:PRINT'ВВЕДИТЕ RE B('I')'
80 INPUT C(I):PRINT'ВВЕДИТЕ IM B('I')'
90 INPUT D(I):NEXT I
100 FOR I=1 TO N-1
110 FOR J=I+1 TO N
120 LETH=A(I,I)^2+B(I,I)^2
130 LETX=-(A(J,I)*A(I,I)+B(J,I)*B(I,I))/H
140 LETB(J,I)=-(B(J,I)*A(I,I)-A(J,I)*B(I,I))/H:LETA(J,I)=X
150 FOR K=I+1 TO N
160 LETA(J,K)=A(J,K)+A(J,I)*A(I,K)-B(J,I)*B(I,K)
170 LETB(J,K)=B(J,K)+A(J,I)*B(I,K)+B(J,I)*A(I,K):NEXT K
180 LETC(J)=C(J)+A(J,I)*C(I)-B(J,I)*D(I)
190 LETD(J)=D(J)+A(J,I)*D(I)+B(J,I)*C(I):NEXT J:NEXT I
200 LETH=A(N,N)^2+B(N,N)^2
210 LETK(N)=(C(N)*A(N,N)+D(N)*B(N,N))/H
220 LETL(N)=(D(N)*A(N,N)-C(N)*B(N,N))/H
230 FOR I=N-1 TO 1 STEP -1:LETH=C(I):LETZ=D(I)
240 FOR J=I+1 TO N:LETH=H-K(J)*A(I,J)+L(J)*B(I,J)
250 LETZ=Z-K(J)*B(I,J)-L(J)*A(I,J):NEXT J
260 LETM=A(I,I)^2+B(I,I)^2
270 LETK(I)=(H*A(I,I)+Z*B(I,I))/M
280 LETL(I)=(Z*A(I,I)-H*B(I,I))/M:NEXT I
290 PRINT'КОРНИ СИСТЕМЫ УРАВНЕНИЙ'
300 FOR I=1 TO N:PRINT!2.0!'RE X('I')='!F1.9! K(I)
310 PRINT!2.0!'IM X('I')='!F1.9! L(I):NEXT I:END
```

Пример. Решение систем линейных уравнений с комплексными коэффициентами лежит в основе расчета линейных цепей, содержащих реактивные элементы, на переменном токе. Комплексные сопротивления катушки индуктивности $\dot{X}_L$ и конденсатора $\dot{X}_C$ выражаются как

$$\dot{X}_L = i\omega L \quad \text{и} \quad \dot{X}_C = \frac{1}{i\omega C} = -\frac{i}{\omega C},$$

где $\omega = 2\pi f$ – круговая частота; f – частота синусоидального тока; L – индуктивность; C – емкость.

Пусть необходимо вычислить комплексные токи $\dot{I}_1$ и $\dot{I}_2$ в цепи на рис. 3.4. Для этой цепи приведены значения сопротивлений (активные и реактивные) при некоторой частоте f генератора ЭДС переменного тока. Для нахождения токов $\dot{I}_1$ и $\dot{I}_2$ нужно составить систему из двух линейных уравнений

$$(10 + i\ 200)\dot{I}_1 - i\ 200\dot{I}_2 = 5,$$
$$-i\ 200\dot{I}_1 + (200 - 30)i\dot{I}_2 = 0.$$

В матричной форме она имеет вид

$$\begin{vmatrix} 10 + i\ 200 & -i\ 200 \\ -i\ 200 & i\ 170 \end{vmatrix} \cdot \begin{vmatrix} \dot{I}_1 \\ \dot{I}_2 \end{vmatrix} = \begin{vmatrix} 5 \\ 0 \end{vmatrix}.$$

Таким образом, $N = 2$, $a_{11} = 10$, $b_{11} = 200$, $a_{12} = 0$, $b_{12} = -200$, $a_{21} = 0$, $b_{21} = -200$, $a_{22} = 0$, $b_{22} = 170$, $c_1 = 5$, $d_1 = 0$, $c_2 = 0$ и $d_2 = 0$. Введя эти данные в программы, приведенные выше (по соответствующему запросу в ходе диалога), получим:

$I_1 = X_1 = 0{,}037156081 + J * (0{,}13113911)$ (А)
$I_2 = X_2 = 0{,}043713037 + J * (0{,}154281131)$ (А)

(мнимая единица $i = \sqrt{-1}$ в выводе результатов обозначена как J).

3.3. ОСОБЕННОСТИ РАСЧЕТОВ НА ПЭВМ С МАЛОЙ ЕМКОСТЬЮ ОЗУ (ОПЕРАТОР DEFM)

Карманные ПЭВМ и ПЭВМ-микрокалькуляторы имеют ОЗУ малой емкости (обычно от 1 до 4 Кбайт). В этом случае выполнение операций с массивами имеет свою специфику.

У таких ПЭВМ ОЗУ имеет упрощенную структуру с жесткой привязкой соответствующих областей к именам переменных. Например, ОЗУ ПЭВМ-микрокалькулятора FX-702P имеет следующую структуру:

A, B, . . . , Z – 26 переменных

A∅	A1	A2	. . .	A9	Двухмерный массив 20×10 переменных A (I, J) или одномерный из 200 переменных A (I)
B∅	B1	B2	. . .	B9	
. . .	. . .	. . .	. . .	. . .	
T∅	T1	T2	. . .	T9	

26 переменных (от A до Z) всегда находятся в распоряжении пользователя. Область ОЗУ с двухмерным или одномерным массивом перераспределяется между данными и шагами программы. Если вся эта область (200 переменных) отведена под данные, то программа может содержать всего до 80 шагов. Сокращая число переменных на каждый десяток, можно увеличить число программных шагов на 80. Для этого используется оператор DEFM n, где $n = 0$–19 – число, на 1 меньшее числа десятков регистров (переменных), отведенных под программу. Если вся емкость ОЗУ отведена под программу, число шагов ее возрастает до 1680.

Применение переменных вида A2, B9, T3 и т.д. в данном случае ограничено. Их задание ведет ко входу в единственный двухмерный массив и исключает применение его для других расчетов.

Двухмерный массив задается в виде A (α, β), где A – имя массива; α, β – арифметические выражения, задающие число строк (от 0 до 19) и число столбцов (от 0 до 9). При таком представлении нельзя задавать массив какой-либо другой буквой. Например, выражения B (5, 9) или C (I, J) являются ошибочными. Можно задавать лишь A (5, 9), A (I, J) и т.д. Очевидно, что при этом A (∅, ∅) = A∅, A (∅, 9) = A9, A (I, ∅) = = B∅, . . . , A (19, 9) = T9. Это в принципе отличает данную версию Бейсика от общепринятой, где A (∅, ∅) и A∅ или A (19,9) и T9 являются совершенно различными и не зависящими друг от друга переменными.

Возможно и задание одномерного массива A (α), где α – арифметическое выражение, задающее номер элемента массива (от 0 до 199). Здесь также недопустимы иные

(кроме A) имена, например B(2), C(9) и т.д. Имеет место соответствие: A(Ø) = AØ, A(1) = A1, . . . , A(1Ø) = BØ, A(11) = B1, . . . , A(199) = T9.

Малая емкость ОЗУ должна учитываться и при составлении программ. Каждый номер строки (от 1 до 9999) занимает два шага программы, каждый оператор Бейсика – один шаг и каждый знак (включая пробел) также один шаг. На завершение ввода каждой строки нажатием клавиши EXE также затрачивается один шаг программы.

Исходя из сказанного, следует учитывать некоторые общие рекомендации по экономии емкости ОЗУ:

1. Целесообразно включение в строку нескольких инструкций с целью уменьшения числа номеров строк и ввода команд EXE.

2. Использовать по возможности краткие комментарии, например не INPUT X = (Введите X =), а X = ? или просто X? и т.д.

3. Широко применять циклы, подпрограммы и встроенные микропрограммы для сокращения общей длины программы.

4. Несколько массивов можно формировать из одного A(I) или A(I, J), разбив его на части. Например, 10 одномерных массивов по 10 элементов можно получить из одного массива A(I) с I = 0, 1, . . . , 99, представив их в виде A(I), A(10 + I), A(20 + I), . . . , A(90 + I).

5. Тщательно оптимизировать программу (см. § 3.5).

В качестве примера применения этих рекомендаций рассмотрим программу для решения систем из $N \leqslant 9$ линейных уравнений методом Гаусса на ПЭВМ FX-702P:

```
10 PRT "METHOD GAUSS": INP "N=",N
20 FOR I=1 TO N: FOR J=1 TO N
30 PRT "A";I;",";J;"=";: INP A(I,J)
40 IF J=N: PRT "B";I;"=";: INP A(I,Ø)
50 NEXT J: NEXT I
60 FOR I=1 TO N-1: FOR J=I+1 TO N
70 A(J,I)=-A(J,I)/A(I,I): FOR K=I+1 TO N
80 A(J,K)=A(J,K)+A(J,I)*A(I,K): NEXT K
90 A(J,Ø)=A(J,Ø)+A(J,I)*A(I,Ø)
100 NEXT J: NEXT I
110 A(Ø,N)=A(N,Ø)/A(N,N)
120 FOR I=N-1 TO 1 STEP -1: H=A(I,Ø)
130 FOR J=I+1 TO N: H=H-A(Ø,J)*A(I,J): NEXT J
140 A(Ø,I)=H/A(I,I): NEXT I
150 FOR I=1 TO N
160 PRT "X";I;"=";A(Ø,I): NEXT I
```

В строках 20–50 этой программы организован ввод коэффициентов $a_{11} - a_{nn}$ и $b_1 - b_n$. Под $b_1 - b_n$ выделен массив A(I, Ø), под $a_{11} - a_{nn}$ – массив A(I, J) при I = 1, 2, . . . , N и J = 1, 2, . . . , N. В строках 60–100 реализован прямой ход метода Гаусса. В конце его получается значение x_n. Под неизвестные $x_1, x_2, \ldots, x_n$ отведен массив A(Ø, I). В строках 110–140 организован обратный ход метода Гаусса, в строках 150 и 160 – вывод значений $x_1, x_2, \ldots, x_n$.

Рассмотрим решение по этой программе системы из трех линейных уравнений

$$\begin{bmatrix} 4 & 0{,}24 & -0{,}08 \\ 0{,}09 & 3 & -0{,}15 \\ 0{,}04 & -0{,}08 & 4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 9 \\ 20 \end{bmatrix}.$$

Диалог с ПЭВМ происходит следующим образом (EXE – клавиша фиксации ввода):

Индикация	Ввод	Комментарий
METOD GAUSS	F_1 9	Пуск, выдача названия программы
	CONT	Продолжение вычислений
N?	3 EXE	Ввод N
A1.1 ?	4 EXE	Ввод a_{11}
A1.2 ?	24 EXE	Ввод a_{12}
A1.3 ?	–.Ø8 EXE	Ввод a_{13}
B1 ?	8 EXE	Ввод b_1
A2.1 ?	.09 EXE	Ввод a_{21}
A2.2?	3 EXE	Ввод a_{22}
A2.3?	–.15 EXE	Ввод a_{23}
B2?	9 EXE	Ввод b_2
A3.1?	.Ø4 EXE	Ввод a_{31}
A3.2?	–.Ø8 EXE	Ввод a_{32}
A3.3?	4 EXE	Ввод a_{33}
B3?	2Ø EXE	Ввод b_3
X1 = 1.9Ø9198281		Вывод x_1
X2 = 3.194964417		Вывод x_2
X3 = 5.Ø4480736		Вывод x_3
		Конец вычислений (мигающая черта-курсор)

Перевод программ с версии Бейсика ПЭВМ с ОЗУ малой емкости на версии Бейсика более мощных ПЭВМ возможен дословно. Исключение составляют лишь программы, содержащие специфичные операторы, например статистические (см. § 3.4) или преобразования углов, вычисления факториала. Современные ПЭВМ-микрокалькуляторы обладают большим числом подобных операторов, что частично компенсирует малую емкость ОЗУ. При обратном переводе надо учитывать приведенные выше рекомендации, особенно касающиеся возможной организации массивов.

Некоторые простые ПЭВМ (например, Aquarius), не имеют операторов для вычисления обратных тригонометрических функций. С погрешностью менее 0,0006° функция arctg x вычисляется с помощью известной аппроксимации

$$\varphi^\circ = \operatorname{arctg} x \approx 90\,x/\sqrt{1{,}21163 + x^2 + \sqrt{1{,}5762 + 1{,}6223x^2}}.$$

Это вычисление реализуется на Бейсике следующим предложением (φ° = F)

```
F = 9Ø * X/SQR(1.21163 + X * X + SQR(1.5762 + 1.6223 * X * X))
```

Реальная погрешность (с учетом погрешностей усечения чисел в ПЭВМ) может несколько превышать указанную.

Остальные обратные тригонометрические функции вычисляются по формулам:

$$\arcsin x = \operatorname{arctg}\left(x/\sqrt{1-x^2}\right);$$
$$\arccos x = \pi/2 - \operatorname{arctg}\left(x/\sqrt{1-x^2}\right) = \pi/2 - \arcsin x;$$
$$\operatorname{arcctg} x = \pi/2 - \operatorname{arctg} x.$$

3.4. СТАТИСТИЧЕСКИЕ РАСЧЕТЫ НА СПЕЦИАЛИЗИРОВАННЫХ ПЭВМ (ОПЕРАТОРЫ STAT, ASTAT, SAC, ФУНКЦИИ SDX, SDY, SDXN, SDYN, LRA, LRB, COR, EOX, EOY, MX, MY, SX, SY, SX2, SY2, SXY)

ПЭВМ-микрокалькуляторы, ориентированные на инженерные и научные расчеты, могут микропрограммно выполнять многие типовые статистические расчеты: вычисление среднего и дисперсии, взаимной корреляции, коэффициентов линейной регрессии и др. Так, ПЭВМ-калькулятор FX-702P (и другие подобные модели) имеет следующие операторы для статических расчетов (статистики):

SAC – очищение (обнуление) всех служебных регистров статистических расчетов,

STAT X – ввод x_i при одномерной статистике,

STAT N ; X – ввод групповых данных x_i (N – число x_i в группе),

STAT X, Y – ввод пары x_i, y_i отсчетов при двухмерной статистике,

STAT N ; X, Y – ввод групповых данных при двухмерной статистике (N число пар x_i, y_i, X и Y – средние значения x_i и y_i),

CNT – вывод числа введенных отсчетов,

SX, SY – вычисление сумм $\sum_{i=1}^{n} x_i$, $\sum_{i=1}^{n} y_i$,

SX2, SY2 – вычисление сумм $\sum_{i=1}^{n} x_i^2$, $\sum_{i=1}^{n} y_i^2$,

SXY – вычисление суммы $\sum_{i=1}^{n} x_i, y_i$,

MX, MY – вычисление среднего $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$,

SDX – вычисление среднеквадратического отклонения несмещенного

$$\sigma_{n-1,x} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2/n}{n-1}},$$

SDY – вычисление среднеквадратического отклонения несмещенного

$$\sigma_{n-1,y} = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}} = \sqrt{\frac{\sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2/n}{n-1}},$$

SDXN – вычисление стандартного среднеквадратического отклонения смещенного

$$\sigma_{n,x} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2},$$

SDYN – вычисление стандартного отклонения смещенного

$$\sigma_{n,y} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2},$$

LRA – вычисление коэффициента линейной регрессии $y(x) = A + Bx$ методом наименьших квадратов $A = (\sum_{i=1}^{n} y_i - B \sum_{i=1}^{n}) / n$,

LRB – вычисление коэффициента линейной регрессии методом наименьших квадратов

$$B = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2},$$

COR – вычисление взаимной корреляции

$$r = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{[n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2][n \sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2]}},$$

EOX – вычисление x по заданному значению y из выражения $x = (y - A)/B$,
EOY – вычисление y по заданному x из выражения $y = A + B$,
ASIAT – автоматическая выдача всех параметров.

С помощью этих статистических параметров можно легко найти и другие: несмещенные дисперсии $D_{0,x} = \sigma^2_{n-1,x}$ и $D_{0,y} = \sigma^2_{n-1,y}$, смещенные дисперсии $D_x = \sigma^2_{n,x}$ и $D_y = \sigma^2_{n,y}$, коэффициенты вариации $\upsilon_x = D_{0,x}/\overline{x}$ и $\upsilon_y = D_{0,y}/\overline{y}$ и т. д.

Описанные операторы могут использоваться как в режиме калькулятора, так и в режиме вычислений по программе. В последнем случае они существенно сокращают длину программ для статистических расчетов. В качестве примера ниже приведена программа экспоненциальной регресии для ПЭВМ-микрокалькулятора FX-702P:

```
10 SAC: INP "N=",N
20 FOR I=1 TO N
30 INP "X=",X,"Y=",Y
40 STAT X,LNY: NEXT I
50 PRT "A=";EXP LRA,"B=";LRB,"R=";COR
60 END
```

С помощью этой программы по ряду значений x_i и y_i вычисляются параметры A и B экспоненциальной зависимости $y(x) = A\exp(Bx)$. Ее можно записать в виде $\ln y(x) = \ln A + Bx$. Следовательно, эта зависимость сводится к линейной, если вместо y_i задавать $\ln y_i$ (см. начало программы), а затем вычислять $A = \exp A'$ и $B = B'$, где A' и B' – параметры линейной регрессии.

Допустим зависимость x_i (y_i) задана числами 6,9 (21,4), 12,9 (15,7), 19,8 (12,1), 26,7 (8,5) и 35,1 (5,2). Запустив программу по запросу, вводим $N = 5$ и значения x_i и y_i. После этого программа вычисляет $A = 30{,}49758742$, $B = -0{,}04920370831$ и $r = -0{,}9972473519$.

Программы на Бейсике для других видов зависимостей $y(x)$, включая полиномиальную аппроксимацию, можно найти в [1].

3.5. РАБОТА В КОДАХ (ОПЕРАТОРЫ OUT, POKE, USR, CALL, ФУНКЦИИ IN, CODE, PEEK, CHR$, ASC)

Большинство ПЭВМ имеет специальные команды для входа непосредственно в ОЗУ: ввода в ОЗУ команд в кодах и вызова кодов, хранящихся в ОЗУ. Как правило, большинство задач может решаться без применения этих конструкций – достаточно пользоваться обычными командами Бейсика. Однако в ряде случаев возможность работы с кодами весьма полезна.

С помощью инструкции в кодах можно получить некоторые специальные эффекты, например вывести особые знаки на экран дисплея, изменить тембр щелчка, имитирующего нажатие клавиш, вывести информацию на индикацию в особые (обычно недоступные) места экрана дисплея и т.д. Ввод команд прямо в ОЗУ происходит быстрее, чем при использовании обычных инструкций программы – интерпретатора языка Бейсик, поскольку последние интерпретируются в первую очередь. Наконец, такие команды обеспечивают возможность перехода в режим программирования на языке ассемблера данной ПЭВМ, а иногда обеспечивают и доступ непосредственно к микропроцессору.

Как отмечалось (см. § 1.2), ОЗУ содержит большое число ячеек, каждая из которых имеет свой адрес, указываемый десятичным или шестнадцатеричным числом. В каждую ячейку может помещаться код-инструкция (также десятичное или шестнадцатеричное число).

Ввод кода в адресуемую ячейку ОЗУ или порт. Для ввода кода в ячейку ОЗУ с заданным адресом используются операторы

OUT Адрес, Код
POKE Адрес, Код

Оператор OUT обычно оперирует с шестнадцатеричными числами и применяется для указания адреса порта, с которого поступает информация (код), команда POKE – с десятичными числами, засылает число-код по указанному адресу. Например, POKE 1ØØØØ, 65 засылает код 65 в ячейку ОЗУ с адресом 10000. Число свободных ячеек ОЗУ выдает функция FRE.

Вывод кода из адресуемой ячейки ОЗУ или порта. Обеспечивается функциями

IN Адрес
PEEK Адрес

Например, команда-функция PEEK 1ØØØØ в предшествующем случае создает код 65. Таким образом, если задать исполнение команды PRINT PEEK 1ØØØØ, на индикацию поступит число 65. Команда LET C= =PEEK 1ØØØØ означает присвоение переменной C значения 65. Действие функции PEEK иллюстрирует следующая программа:

```
10 REM ФУНКЦИЯ PEEK (АДРЕС)
20 FOR A=23760 TO 23764
30 PRINT "АДРЕС=";A,"КОД=";PEEK (A)
40 NEXT A
```

```
АДРЕС=23760      КОД=149
АДРЕС=23761      КОД=164
АДРЕС=23762      КОД=72
АДРЕС=23763      КОД=75
АДРЕС=23764      КОД=146
```

Преобразование знаков в код. Символы (знаки) можно преобразовать в стандартный код с помощью команд

CODE "Знак"
ASC ("Знак")

Обычно применяется одна из них (в зависимости от версии языка Бейсик). Например, для ПЭВМ ZX-Spectrum функция CODE "A" дает значение кода 65 буквы A, для ПЭВМ IBM PC функция ASC ("R") значение кода 82 буквы R. Знак может задаваться значением символьной переменной, причем воспринимается только ее первый знак. Например, если A$ = "RADAR", то выполнение команды PRINT CODE A$ или PRINT ASC (A$) выводит на индикацию число 82 – код буквы R.

Программа, приведенная ниже, позволяет наблюдать коды любого знака, для этого достаточно нажать клавишу с этим знаком, а затем клавишу ENTER:

```
10 REM ДЕЙСТВИЕ ФУНКЦИИ CODE
20 INPUT A$
30 PRINT A$,"КОД=";CODE A$
40 GO TO 20
```

```
1                КОД=49
2                КОД=50
3                КОД=51
A                КОД=65
B                КОД=66
J                КОД=74
```

Преобразование кодов в знаки и слова. Код может быть преобразован в знак (иногда в целое слово инструкции) с помощью функции

CHR$(¤) Код

Например, операция PRINT CHR¤ 65 выводит на индикацию букву A, код которой 65:

```
10 REM ДЕЙСТВИЕ ФУНКЦИИ CHR¤ (КОД)
20 FOR A=63 TO 68
30 PRINT "КОД=";A,"СИМВОЛ  ";CHR¤ (A)
40 NEXT A

КОД=63        СИМВОЛ  ?
КОД=64        СИМВОЛ  @
КОД=65        СИМВОЛ  A
КОД=66        СИМВОЛ  B
КОД=67        СИМВОЛ  C
КОД=68        СИМВОЛ  D
```

У некоторых ПЭВМ (например, ZX-Spectrum) закодированы служебные слова языка Бейсик. К примеру, операция PRINT CHR$ 249 выводит на индикацию слово RANDOMIZE, имеющее код 249.

Обращение к подпрограммам пользователя. Во многих версиях Бейсика предусмотрено обращение к подпрограммам пользователя, записываемым на языке машинных кодов, ассемблера или других алгоритмических языках. Обращение такой подпрограмме обычно задается оператором

[НС] USR [*N*]

где *N* – номер подпрограммы или адрес ее начала в ОЗУ; модификатор USR указывает на обращение к подпрограмме на другом языке. Например, в ПЭВМ ZX-Spectrum обращение к подпрограмме может задаваться в виде

[НС] RANDOMIZE USR *N*

У некоторых ПЭВМ (например, IBM PC), обращение к подпрограммам пользователя выполняется с помощью оператора CAL(*N*), где *N* задает начальный адрес подпрограммы.

Ввод команд на языке ассемблера или в машинных кодах. Подпрограммы, записываемые в машинных кодах, можно вводить в ПЭВМ, используя оператор POKE. Для этого необходимо выделить область в ОЗУ за пределами областей, занятых под другие функции, например под программу на языке Бейсик. Если в ходе ввода бейсик-программы (или ее выполнения) выделенная область ОЗУ захватывается, это неизбежно приводит к нарушению программы, введенной на языке ассемблера или в машинных кодах, и исключает правильную ее работу. При этом возможны самые неожиданные и неопределенные ситуации, например отказ от выполнения любых команд ("зависание" ПЭВМ). Вывести ПЭВМ из такого крайне неприятного состояния нередко удается единственным способом – выключив питание и потеряв введенные данные и программу.

Как упоминалось, программа на языке ассемблера обычно записывается в виде мнемокодов. Для их перевода в машинные коды используется довольно сложная системная программа – *ассемблер*. Иногда (очень редко) она встроена в ПЭВМ, но обычно поставляется записанной на магнитную ленту или диск. Однако пользователь, не имеющий такой программы, может легко ввести в свои бейсик-программы программы, записанные на языке ассемблера. При этом ему придется вручную выполнить трансляцию мнемокодов ассемблера в машинные коды ПЭВМ. Поясним это простым примером.

Пусть требуется организовать выполнение на ассемблере следующих операций (для ПЭВМ с микропроцессорами класса 8080 или Z=80): ввести число 0074 в пару регистров B и C, увеличить это число на 1 и обеспечить возврат из подпрограммы. На ассемблере эти операции записываются следующим образом:

LD BC, ØØ,74 – ввод числа 0074 в регистры B и C,
INC BC – увеличение (инкремент) содержимого регистров B и C на 1,
RET – возврат из подпрограммы.

Для ввода этой подпрограммы нужно выполнить следующие подготовительные работы: выделить область ОЗУ, указать номер начальной ячейки ОЗУ, с которой идет ввод подпрограммы, преобразовать мнемокоды ассемблера в машинные коды (деся-

тичные, шестнадцатеричные или двоичные), ввести машинные коды с помощью операторов POKE.

Допустим, рассмотренная программа будет вводиться в ОЗУ, начиная с адреса 35000. Тогда в полном виде запись программы имеет вид

Адрес ОЗУ	Мнемокод ассемблера	Код		
		десятичный	шестнадцатеричный	двоичный
35000	LD BC	1	1	00000001
35001	74	74	4A	01001010
35002	0	0	00	00000000
35003	INC BC	3	03	00000011
35004	RET	201	C9	11001001

Вторая колонка (мнемокоды) фактически в ПЭВМ не вводится, она нужна пользователю лишь для лучшего восприятия программы. Три последние колонки (с кодами) также полностью не используются ПЭВМ (в зависимости от ее типа), достаточно ввести лишь коды одного из указанных трех типов. Если, например, ПЭВМ допускает ввод десятичных кодов (самый распространенный случай), то ввод указанной подпрограммы в простейшем случае сводится к исполнению следующих команд:

```
POKE  35000,  1      Ввод команды LD BC
POKE  35001,  74 }
POKE  35002,  0  }   Ввод числа 74 в виде 0074
POKE  35003,  3      Ввод команды INC BC
POKE  35004,  201    Ввод команды RET
```

В этом случае не требуется никакой дополнительной программы для ввода машинных кодов. Если коды заданы в виде двоичных или шестнадцатеричных чисел, нужно преобразовать их в десятичные коды с помощью специальных операторов или простых программ. Например, ПЭВМ ZX-Spectrum имеет специальный оператор BIN для преобразования двоичных чисел в десятичные. В эту ПЭВМ данная подпрограмма может вводиться с применением двоичных кодов:

```
POKE  35000,  BIN  00000001
POKE  35001,  BIN  01001010
POKE  35002,  BIN  00000000
POKE  35003,  BIN  00000011
POKE  35004,  BIN  11001001
```

Нетрудно заметить, что ввод в двоичных кодах более громоздок, его целесообразно использовать, если исходная программа задана в двоичных кодах. Шестнадцатеричные двухбайтовые коды переводятся в десятичные с помощью формулы

$$K_{10} = K_{16}^{(1)} \cdot 16 + K_{16}^{(2)},$$

где $K_{16}^{(1)}$ – численное (десятичное) значение первого шестнадцатеричного символа, а $K_{16}^{(2)}$ – второго символа. Например, для шестнадцатеричного кода C9 команды RET $K_{10} = 12 \cdot 16 + 9 = 201$.

Для ввода больших подпрограмм на языке ассемблера, заданных большими массивами кодов, удобно применять специальные относительно простые программы. Например, ввод с применением десятичных машинных кодов можно обеспечить с помощью программы

```
10 INPUT "ВВЕДИТЕ НАЧАЛЬНЫЙ АДРЕС ";A
20 INPUT "ВВЕДИТЕ КОД ";C: POKE A,C
30 PRINT "A=";A,"C=";C
40 LET A=A+1
50 GOTO 20
```

В строке 10 задается начальный адрес – значение переменной A, затем (строка 20) код C, он вводится в ячейку ОЗУ с адресом A с помощью оператора POKE. В строке 30 задана печать адресов и кодов по мере ввода последних. В строке 40 адрес A получает приращение на 1, а в строке 50 обеспечивается безусловный переход к строке 20, т.е. вводу нового десятичного кода.

Однако при большом массиве машинных кодов их ввод утомителен и чреват ошибками. Как отмечалось, они крайне нежелательны, так как могут приводить к непредсказуемому поведению ПЭВМ. Поэтому рекомендуются меры по уменьшению таких ошибок. Широко применяется, например, задание кодов по группам (из 5 или 10 кодов) с указанием и контролем их суммы. Описанный прием иллюстрирует программа

```
10 LET A=35000
20 LET S=0
30 FOR I=0 TO 5
40 READ C: POKE A+I,C
50 LET S=S+C
60 NEXT I: READ S1
70 IF S<>S1 THEN PRINT "ОШИБКА ВВОДА КОДОВ": STOP
80 LET A=A+5: GOTO 20
1000 DATA 66,74,0,3,201,344
.........................
```

В строке 10 задано начальное значение адреса (A=35000). В строке 20 обнуляется переменная суммы кодов S. В строках 30 и 40 задан ввод пяти кодов (их значения присваиваются переменной C). В строке 40 вычисляется текущая сумма кодов S. В строке 60 задается выход из цикла и присвоение переменной S1 значения, равного сумме кодов.

Таблица кодов задается в поле оператором DATA. В каждой строке задается пять кодов и шестое число – их сумма (см. пример в строке 1000). В строке 70 сумма первых пяти чисел (кодов) сравнивается с шестым числом. Если какой-либо код набран неверно, печатается сообщение ОШИБКА ПРИ ВВОДЕ, что является сигналом к проверке данных (кодов). Отметим, что по значению A легко найти, в какой строке с операторами DATA оказалась ошибка (легко модифицировать программу так, что она будет прямо указывать эту строку). В строке 90 значение A увеличивается на 5 и идет безусловный переход к строке 20, т.е. программа готовится к принятию еще пяти кодов (их записывают после строки 1000).

Задав выполнение описанной подпрограммы, например, вводом команды PRINT USR 35000, получим результат – число 75 (т.е. исходное число

74, увеличенное на 1). Обратим внимание, число 74 задано двумя байтами (второй 74 в ячейке ОЗУ с номером 35001 и первый 00 в ячейке ОЗУ с номером 35002). Таким образом, число 74 задано в виде 0074. Это специфика задания чисел в пары регистров 8-разрядного микропроцессора. Так можно задавать целые числа от 0 до 65535 (всего 2^{16} чисел), они могут быть кодами, данными или адресами. Коды от 0 до 255 можно заносить в одиночные ячейки ОЗУ или одиночные регистры 8-разрядных микропроцессоров.

Системные переменные. У всех ПЭВМ определенная часть ОЗУ выделена под хранение специальных системных переменных, т.е. переменных, значения которых несут информацию о работе ПЭВМ. Они могут указывать, например, в какой области ОЗУ расположена бейсик-программа, какую емкость памяти она занимает, каковы координаты последней построенной точки, какой код последнего индицируемого символа и т.д. – и таких переменных может быть несколько десятков. Зная адреса ячеек ОЗУ, в которых хранятся значения системных переменных, можно прочесть их с помощью команды

PRINT PEEK Адрес

Однако далеко не всегда адреса и перечень системных переменных даются в описаниях ПЭВМ. Системные переменные для удобства запоминания часто задаются *мнемокодами,* как и команды ассемблера. Нельзя путать их имена с инструкциями Бейсика или другого языка программирования высокого уровня.

Часть системных переменных пользователь может менять, воздействуя на работу ПЭВМ. Таким образом можно менять время реакции ПЭВМ на нажатие и отпускание клавиш, вызывать изменение длительности или тона звукового сигнала, получать ряд эффектов, которые нельзя задавать прямо с помощью бейсик-программ. Для изменения значений системных переменных используется команда

POKE Адрес переменной, числовое значение

Следует помнить, что необдуманное и случайное изменение значений системных переменных может резко нарушить нормальную работу ПЭВМ. Именно поэтому системные переменные нередко "секрет фирмы", раскрыть который без должного описания трудно даже опытным пользователям ПЭВМ.

Большое число системных переменных характерно, например, для ПЭВМ ZX-Spectrum, перечень их приводится в техническом описании. Для иллюстрации полезности применения их приведем несколько примеров.

Команда POKE 23609, *n* (здесь $n = 0 - 255$) позволяет менять длительность звукового сигнала, формируемого при нажатии любой клавиши, она растет с ростом *n*. Команда POKE 23692, 255 устраняет присущую ПЭВМ ZX-Spectrum автоматическую остановку вычислений при заполнении всего экрана данными. Команды PEEK 23672, PEEK 23673 и PEEK 23674 генерируют числа от 0 до 255 через каждые 20, 20·256 и

$20 \cdot 256 \cdot 256$ мс. Их можно использовать для создания высокоточных часов (погрешность около 0,01 %). Функция POKE 22528 + *i*, *c* (здесь *i* – номер знакоместа от 0 до 767 и *c* – код знака) выводит знак с кодом *c* на любое *i*-е знакоместо.

Системные функции часто задаются с помощью операторов CHR$ и PRINT. Например, в версиях Бейсика фирмы Microsoft функция CHR$ (11) используется для очистки экрана. Ряд таких функций применяется для задания положения маркера-курсора и управления принтером. Грамотное владение системными функциями весьма полезно при составлении сложных, например обучающих и игровых, программ.

3.6. СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ (ОПЕРАТОРЫ DEF PROC, END PROC, LOCAL, DEFAULT, DO, LOOP, REPEAT)

Большинство версий Бейсика, как отмечалось, плохо приспособлено к структурному программированию. У них отсутствуют операторы задания поименованных процедур, локализации в них переменных и организации ряда специальных циклов. Однако эти недостатки устранены в некоторых современных расширенных версиях Бейсика (Beta-BASIC 1.8, Beta-BASIC 3.0, Бейсик ПЭВМ НР-85 и др.) [54–56].

Процедура – особый вид подпрограммы, имеющей имя и входные параметры. Обращение к процедуре (в отличие от обращения GOSUB НС к обычной подпрограмме) производится по имени. Процедура может стоять в любом месте программы как до ее вызова, так и после. В первом случае она игнорируется. Процедура обычно имеет следующую структуру:

```
DEF PROC Имя [(Список переменных)]
         Тело процедуры
END PROC
```

Например, следующая процедура:

```
1Ø DEF PROC DEMO
2Ø PRINT "HELLO"
3Ø END PROC
```

задает печать слова HELLO (Привет). В данном случае в заголовке процедуры DEF PROC имеется только ее имя DEMO. Входных параметров в виде списка переменных нет.

Обращение к процедуре осуществляется с помощью команды

```
PROC Имя [(Значения переменных)]
```

Так, описанная выше процедура выдает на печать слово HELLO, если задать обращение PROC DEMO.

Процедура может выполнять роль функции, если она вырабатывает некоторое значение одного выходного параметра. Описанная только что процедура может рассматриваться как функция, вырабатывающая слово ПРИВЕТ. В некоторых версиях Бейсика вызов процедур-функций может выполняться просто указанием имени – в нашем случае DEMO. В общем случае процедура может иметь несколько выходных параметров или не иметь их.

Применение процедуры вычисления модуля $M=\sqrt{X^2+Y^2}$ комплексного числа $Z=X+iY$ с помощью процедуры MODUL X, Y с двумя входными параметрами – переменными X и Y иллюстрирует программа:

```
  5 REM DEFINED PROCEDURE
 10 DEF PROC MODUL X,Y
 20   LET M=SQR (X*X+Y*Y)
 30 END PROC
 95 REM BASIC PROGRAMM
100 INPUT "ENTER X Y ";X,Y
110 PROC MODUL X,Y
120 PRINT "M=";M
130 STOP
```

При пуске программы командой RUN строки 5–95 пропускаются и исполняется строка 100 – ввод значений X и Y. Затем в строке 110 происходит обращение к процедуре MODUL X, Y. При этом выполняются строки 10–30, т.е. переменной M присваивается значение модуля Z. К примеру, если задать $X=4$ и $Y=3$, получим $M=5$.

Заменим строки 100–130 следующими:

```
  5 REM DEFINED PROCEDURE
 10 DEF PROC MODUL X,Y
 20   LET M=SQR (X*X+Y*Y)
 30 END PROC
 95 REM BASIC PROGRAMM
110 PROC MODUL 4,3
120 PRINT "M=";M
130 STOP
```

Результат выполнения этой программы будет прежним. Однако в этом случае в процедуре MODUL X, Y вместо имен переменных X и Y стоят их числовые значения (входные параметры). Отметим, что входными параметрами могут быть и арифметические выражения. Например, правильно обращение к процедуре вида PROC MODUL 2*2, 1.5+1.5.

Глобальными переменными называются переменные, доступные в любой области программы. В обычных версиях Бейсика все переменные являются глобальными, и на это обычно не обращают особого внимания. Будучи где-то определенной (т.е. получив значение), глобальная переменная всюду сохраняет это значение до нового определения. Определение, в свою очередь, может быть в любой части программы, в том числе в процедуре. Так, в рассмотренной выше программе все переменные (X, Y и M) являются глобальными. Переменные X и Y определены в основной программе (строка 100), но используются как определенные переменные в процедуре. Переменная M определяется в процедуре (строка 20), а затем используется в строке 120 основной программы.

В программах, реализующих методы структурного программирования, глобализация переменных лишает процедуры полной самостоятельности, часто она и не нужна. Более того, нередко изменение значений переменной в процессе исполнения процедуры даже необходимо. Однако в общем необходимо, чтобы переменные в процедуре были *локальными*, т.е. их определения внутри процедуры действовали только в ней. Это позволяет избежать часто нежелательных побочных эффектов.

Локализация переменных может производиться двумя способами: использованием их в списке входных параметров процедур и приданием свойства локальности с помощью оператора

LOCAL ⟨Список имен переменных⟩

Первый способ локализации поясняет программа вычисления гиперболического синуса $\mathrm{sh}(x) = (e^x - e^{-x})/2$ с соответствующей процедурой

```
  5 REM DEFINED PROCEDURE
 10 DEF PROC HSN X
 20    LET HSN=(EXP (X)-EXP (-X)
)/2
 30 END PROC
100 REM ENTER GLOBAL VARIABLE X
=15
110 LET X=15
120 REM CALCULATION HSN(1)
130 PROC HSN 1
140 REM OUTPTT HSN(X)
150 PRINT "HSN=";HSN,"X=";X
160 STOP
```

В строке 100 глобальной переменной присвоено значение X = 15. В строке 130 идет обращение к процедуре HSN с аргументом X = 1. Однако теперь та же переменная оказывается локальной, т. е. значение X = 1 действует лишь пока выполняется процедура HSN, на печать (строка 150) выводится значение X

HSN = 1.1752Ø12 X = 15

как глобальной переменной.

Из этого примера видно, что глобальные и локальные переменные могут иметь одни и те же имена, но ведут себя по-разному. Указанное обстоятельство облегчает работу программиста по поиску подходящих имен для переменных. Однако не рекомендуется давать одинаковые имена глобальным и локальным переменным, так как их различное поведение сильно затрудняет анализ программ и чревато непредвиденными последствиями. Одни и те же имена могут иметь как переменные, так и процедуры (в нашем случае HSN).

С помощью оператора

LOCAL ⟨Список имен переменных⟩

можно придать локальный характер переменным, указанным в списке. Действие оператора LOCAL выявляет следующая программа:

```
 10 LET N=123
    LET S=456
 20 DEF PROC CICLE
 30    LOCAL N
       LET S=Ø
 40    FOR N=1 TO 5
 50       LET S=S+N
 60    NEXT N
 70    PRINT "S=";S,"N=";N
 80 END PROC
 90 CICLE
100 PRINT "S=";S,"N=";N
110 STOP
```

Вначале (строка 10) переменным N и S присвоены значения 123 и 456. Далее (строка 30) переменная N объявлена локальной для процедуры с именем CICLE. В строках 40–80 задано вычисление суммы чисел N = 1, 2, 3, 4 и 5 с помощью цикла FOR–NEXT. На печать в строке 70 процедуры выводятся значения

S = 15 N = 6

после обращения к процедуре CICLE в строке 90. Затем значения этих переменных выводятся повторно:

S = 15 N = 123

Итак, глобальная переменная S изменила свое значение в процедуре, а локальная переменная N сохранила свое первоначальное значение как значение глобальной переменной.

Специальный оператор первого присваивания

DEFAULT Имя переменной = значение

обеспечивает присвоение переменной указанного значения, если ранее оно не производилось оператором LET. В противном случае действие оператора DEFAULT игнорируется. Например:

```
10 LET X=1
20 DEFAULT X=5
30 DEFAULT Y=6
40 PRINT "X=";X,"Y=";Y
```

Получим X = 1 (DEFAULT проигнорирован) и Y = 6 (DEFAULT сработал).

Присвоения с помощью оператора DEFAULT (как и LET) в ряде версий Бейсика можно давать в виде

DEFAULT X = 5, Y = 6 и т. д.

В процедурах, заданных операторами DEF PROC и END PROC, можно обращаться к любым другим процедурам и функциям. Однако можно обращаться из процедуры и к самой себе. Такое обращение называется *рекурсией*, а содержащая такое обращение процедура называется *рекурсивной*.

Многие циклические вычисления могут быть организованы с помощью рекурсии. Примером может служить вычисление факториала с помощью следующей рекурсивной процедуры:

```
  10 DEF PROC:N! N,I,F
  20    DEFAULT I=1,F=1
  30    IF I<=N THEN  N! N,I+1,F*
I
        END PROC
     ELSE PRINT "N!=";F
  40 END PROC
```

Отметим несколько особенностей. В списке входных параметров (строка 10) заданы переменные N, I и F. Однако при обращении к процедуре достаточно указать только N. При первом использовании процедуры задается I = F = 1 в строке 20 (применяется оператор DEFAULT). Если I < = N (строка 30), то происходит рекурсивное обращение к процедуре N!, при этом одновременно в списке входных параметров вычисляется значение F*I и I + 1. Таким образом, получаются последовательности значений F 1, 1·2, 1·2·3, . . . , N! и 1, 2, 3, . . . , N!. Если I > N значение F = N! выводится на печать. Обращаясь к процедуре, имеем

N! 0 дает 0! = 1
N! 1 дает 1! = 1
N! 10 дает 10! = 3628800

Рекурсию можно осуществить и с помощью обычных подпрограмм:

```
  10 INPUT "INPUT N=";N
  20 LET I=1: LET F=1: GO SUB 50
  30 PRINT N;"!=";F
  40 GO TO 10
  50 LET F=F*I: LET I=I+1
  60 IF I<=N THEN GO SUB 50: RET
URN
  70 RETURN
```

Здесь рекурсивное обращение в подпрограмме (т. е. к самой себе) содержится в строке 60. Результат работы этой программы подобен приведенному. Следует помнить, что число обращений в данном случае ограничено емкостью специального счетчика подпрограмм.

Рекурсия полезна в ряде специальных вычислений, но там, где можно вполне обойтись без нее, нет смысла ее применять. К примеру, в § 2.10 приведены программы вычисления факториала, которые ничем не уступают приведенным.

Реализация полностью структурированных программ в Бейсике мешает оператор безусловного перехода GOTO НС, однако без него трудно обойтись, особенно если нужен переход вверх. Ухищрения по его замене другими переходами (IF–THEN, например) ведут к потере наглядности программ и их усложнению.

В этой связи наряду с операторами FOR, NEXT, WHILE и UNTIL (см. § 2.10) в версиях Бейсика, ориентированных на структурное программирование, применяют операторы организации циклов DO и LOOP, не содержащие специальных управляющих переменных.

Применение операторов DO и LOOP вместо GOTO иллюстрирует следующая программа:

```
10 DO
20 INPUT "X=";X
30 PRINT "EXP(X)=";EXP (X)
40 LOOP
```

Циклически выполняются все операторы, заключенные между словами DO и LOOP (в данном случае ввод X и вычисление $\exp x$).

Могут строиться также циклы вида:

```
DO WHILE    Условие        DO UNTIL    Условие
    Тело цикла                 Тело цикла
LOOP                       LOOP
```

В цикле с DO WHILE тело цикла выполняется до тех пор, пока условие истинно, а в цикле с DO UNTIL, пока условие ложно. В этих циклах условие проверяется до выполнения тела цикла.

Еще две конструкции циклов типа DO–LOOP:

```
DO                         DO
    Тело цикла                 Тело цикла
LOOP WHILE Условие         LOOP UNTIL Условие
```

аналогичны приведенным выше, но в них условие выполнения цикла проверяется после выполнения тела цикла.

Проиллюстрируем разницу между включением условия в заголовок DO и конец LOOP цикла. Выполнение программы

```
10 LET X=1
20 DO WHILE X<=5
30   PRINT "LN(";X;")=";LN (X)
40   LET X=X+1
50 LOOP
```

выдает на печать (экран дисплея) следующих данных для 5 циклов:

LN(1) =Ø
LN(2) =Ø.69314718
LN(3) =1.Ø986123
LN(4) =1.3862944
LN(5) =1,6Ø94379

Выполнение другого варианта программы

```
10 LET X=1
20 DO WHILE X>=5
30 PRINT "LN(";X;")=";LN (X)
40 LET X=X+1
50 LOOP
```

не ведет к выводу каких-либо результатов, так как условие X>=5 в строке 20 не выполняется. Таким образом, тело цикла ни разу не выполняется.

Наконец, выполнение третьего варианта программы

```
10 LET X=1
20 DO
30 PRINT "LN(";X;")=";LN (X)
40 LET X=X+1
50 LOOP WHILE X>=5
```

дает LN(1) =0. Следовательно, перенос условия WHILE X>=5 из начала в конец цикла повлек за собой выполнение тела цикла один раз, хотя условие X>=5 при этом не выполнялось.

Допускается вложение циклов DO–LOOP друг в друга:

```
DO ┐
DO ┐
LOOP ┘
LOOP ┘
```

```
DO ┐
DO ┐
LOOP ┘
DO ┐
LOOP ┘
LOOP ┘
```

Пересечение циклов недопустимо.

В некоторых версиях встречаются циклы вида REPEAT (повторить) – UNTIL, которые эквивалентны циклам DO–LOOP:

REPEAT	DO
Тело цикла	Тело цикла
UNTIL Условие	LOOP UNTIL Условие
REPEAT	DO
Тело цикла	Тело цикла
UNTIL [FALSE]	LOOP

Число вложений циклов друг в друга ограничено емкостью специального стека, следящего за числом вложений и подпрограмм. Обычно у ПЭВМ несколько десятков вложений.

В ряде случаев возникает необходимость в выходе из тела цикла при исполнении определенного условия. Включение для этого в тело цикла условного перехода обычно не допускается, так как выход по нему нарушает работу специального стека, следящего за правильной очередностью выполнения циклов (см. выше).

В циклах DO–LOOP в тело цикла можно включать специальную команду

EXIT IF Условие

Если заданное условие выполняется, происходит выход из цикла, в противном случае выполнение цикла продолжается. Выход из цикла по этой команде не нарушает работу стека.

Еще одна команда

POP [Имя переменной]

убирает адрес в инструкциях GO SUB, DO–LOOP и PROC. Номер строки становится числовым значением переменной (если ее имя указано). Команда POP позволяет выйти из цикла, подпрограммы или процедуры DEF PROC, не нарушая работу стека из-за утери адреса.

Действие команды POP поясняет такая программа:

```
  10 GO SUB 30
  20 STOP
  30 POP loc
  40 PRINT "SUBROUTINE CALLED FR
OM LINE "; loc
  50 GO TO loc+1
```

Результат ее выполнения следующий:

SUBROUTINE CALLED FROM LINE 10

с остановкой в строке 20 (значение переменной loc = 10 и в строке 50 задан переход к несуществующей строке 11, что ведет к исполнению очередной строки 20 с командой STOP). Таким образом, для современных версий Бейсика нет никаких препятствий для строгой реализации принципов структурного программирования. Поясним сказанное конкретным примером.

Пусть надо создать учебную программу, вычисляющую факториал целых чисел N. Проектирование программы разобъем на ряд этапов.

Этап 1. Составление основной программы. Решение задачи сводится к таким подзадачам: ввод N, установление корректности значений N (для $N < 0$ факториал не вычисляется, а для $N > 33$ у ПЭВМ наступит переполнение разрядной сетки), печать сообщения об ошибках ввода, вычисление $N!$ и печать значения $N!$ Для этого предусмотрим пять частей программы:

1. Основная часть программы.
2. Процедура ввода N(ENTER).
3. Процедура контроля N и вывода сообщений об ошибках ($N < 0$ или $N > 33$) с остановкой счета (CONTROL).

4. Процедура вычисления $N!$ ($N!$). Для разнообразия реализуем вычисления $N! = 1 \cdot 2 \cdot 3 \ldots N$ с помощью цикла DO UNTIL – LOOP.

5. Процедура вывода результата (RESULT).

Листинг программы, составленной по этим правилам:

```
  10 DO
  20   ENTER
  30   CONTROL
  40   N! N
  50   RESULT
  60 LOOP
  70
 100 DEF PROC ENTER
 110   INPUT "INPUT N=";N
 120 END PROC
 130
 200 DEF PROC CONTROL
 210   IF N<0 THEN PRINT "ERROR:
N<0"
         STOP
 220   IF N>33 THEN PRINT "ERROR
 N>33"
         STOP
 230   LET N=INT (N)
 240 END PROC
 250
 300 DEF PROC N! N
 310   LET F=1,I=1
 320   DO UNTIL I=N+1
 330     LET F=F*I,I=I+1
 340   LOOP
 350 END PROC
 360
 400 DEF PROC RESULT
 410   PRINT N;"! =",F
 420 END PROC
```

Хотя программа полностью лишена комментариев, она настолько наглядна, что ее подробное описание лишено смысла. Роль комментариев, по существу, выполняют названия процедур. Однако листинг программы почти втрое длиннее листингов описанных выше программ вычисления факториала. Такова плата за наглядность.

3.7. ДОПОЛНИТЕЛЬНЫЕ ФУНКЦИИ И ОПЕРАТОРЫ (AND, COSE, SINE, OR, RNDM, DEC, MOD, MEM, DPOKE, DPEEK, DEF, KEY, JOIN, KEYWORD, SORT, ITEM, XOR)

В ряде расширений Бейсика встречаются дополнительные функции и операторы [54–56]. Они вводятся указанием комбинации тех или иных клавиш и существенно расширяют возможности языка.

Функция

AND (Целое число 1, Целое число 2)

реализует операцию И над двумя аргументами. Числа анализируются по битам в диапазоне значений от 0 до 65535 (для 8-разрядных ПЭВМ). Если в текущем месте анализируемые биты дают 1, то бит результата будет 1, иначе 0. Так, AND(1,Ø) дает 0, AND(Ø,35) дает 0, AND(35,35) вырабатывает значение 35, а AND(35,1Ø) дает 2. Функция OR реализует подобным образом операцию ИЛИ.

Две функции COSE (аргумент) и SINE (аргумент) аналогичны функциям COS и SIN, но при меньшей точности (четыре верных знака) требуют в 6 раз меньшего времени вычислений. Указанная точность вполне достаточна для многих применений, включая тригонометрические операции графики.

Функция RNDM(Число) служит для быстрого получения случайных чисел – примерно в 2,5 раза быстрее, чем у аналогичной функции RND(Число). Если аргумент (число) задан нулем, то случайные числа генерируются в интервале [0, 1]. Если аргумент задан отличным от 0, то случайные числа генерируются в интервале [0, число]. Так, применяя функцию RNDM(5), будем получать случайные числа в интервале от 0 до 5.

Функция

DEC("Шестнадцатеричное число")

преобразует шестнадцатеричное число в десятичное. Оно лежит в пределах от –255 до +255, если шестнадцатеричное число задано двумя символами, и от –65535 до +65535, если шестнадцатеричное число задано четырьмя символами. Например:

```
PRINT DEC("5")      дает   5
PRINT DEC("11")     дает   17
PRINT DEC("FF")     дает   255
PRINT DEC("4000)    дает   16384
```

Следующие команды:

INPUT A$: POKE address, DEC(A$)

позволяют в ячейку ОЗУ с адресом (значением переменной address) занести десятичный код, соответствующий шестнадцатеричному числу A$, вводимому оператором ввода INPUT.

Обратное преобразование десятичного числа в шестнадцатеричное обеспечивает функция HEX$ (см. § 4.7).

Функция MOD(modulo), представляемая в виде

MOD Число 1, Число 2

создает разность между первым числом и произведением второго числа на наибольшее целое число, обеспечивающее положительную разность, оба числа должны быть положительными. Примеры:

```
PRINT MOD(22.5, 3.5)   дает   1,5
PRINT MOD(100, 33)     дает   1
```

В первом случае имеем $1{,}5 = 22{,}5 - 3{,}5 \cdot 6$, во втором случае $1 = 100 - 33 \cdot 3$.

Функция MEM дает адрес байта в начале свободной области ОЗУ, ее целесообразно использовать для контроля емкости свободной части ОЗУ. Как отмечалось, в одну ячейку (байт) ОЗУ можно поместить десятичный машинный код в виде числа от 0 до 255. Однако у ПЭВМ с 16-разрядной шиной адресов последние лежат в пределах от 0 до 65535. Для хранения таких чисел *N* используются два расположенных рядом байта ОЗУ. Обычно при этом приходится использовать следующие команды:

```
POKE address,   N –INT(N/256) * 256
POKE address, +1,  INT(N/256)
```

Команда "двойное POKE"

DPOKE address, N

обеспечивает размещение числа N в две смежные ячейки ОЗУ, начиная с адреса address, т.е. заменяет указанную цепочку команд. Аналогично функция "двойное PEEK"

DPEEK address

вырабатывает значение целого числа N, хранящегося в двух смежных ячейках ОЗУ с начальным адресом address. Отметим, что адрес может задаваться числом, значением переменной или функции, а также арифметическим выражением. Например, команда

DPOKE 4ØØØØ, 127ØØ

заносит код 12700 в ячейки ОЗУ с адресами 40000 и 40001, а

PRINT DPEEK 4ØØØØ

выдает на печать число 12700, хранящееся в ячейках ОЗУ с адресами 40000 и 40001.

Директива

DEF KEY "Одиночный символ"; "Цепочка символов":

или

DEF KEY Одиночный символ : Цепочка инструкций

задает процедуру, предписанную определенной клавише и вызываемую нажатием ее (точнее, указанием одиночного символа, что может потребовать предварительного нажатия префиксных клавиш). Перед нажатием предопределяющей клавиши нажимаются вместе клавиши SYMBOL SHIFT и SPACE, что вызывает появление знака готовности *. Так, задав процедуру

DEF KEY "Ø" ; "ПРИВЕТ:"

получим вывод слова ПРИВЕТ при нажатии клавиши Ø. В другом случае

```
DEF KEY "T" : INPUT "ВВЕДИТЕ X=" ; X :
PRINT "TANE" = " SINE(X)/COSE(X)
```

при нажатии клавиши T последует запрос

ВВЕДИТЕ X=

После ввода X будет вычислен "быстрый тангенс" в виде значения SINE(X)/COSE(X).

Команда JOIN (соединять) может использоваться как команда соединения строк или объединения переменных и массивов. Например, если имеются массивы A(I) и B(I), то их можно объединить (т.е. массив B(I) включить в массив A(I)) с помощью команды

JOIN B ( ) TO A ( )

При этом массив A(I) расширяется, а массив B(I) исчезает.

Действие оператора JOIN иллюстрирует следующая программа:

```
 10 DIM A(3)
    DIM B(2)
 20 LET A(1)=1,A(2)=2,A(3)=3
 30 LET B(1)=4,B(2)=5
 40 JOIN B() TO A()
 50 FOR I=1 TO 5
 60    PRINT "A(";I;")=";A(I)
 70 NEXT I
 80 PRINT "B(1)=";B(1),"B(2)=";
B(2)
```

Результат ее действия:

A(1) = 1
A(2) = 2
A(3) = 3 } Исходный массив A(I)

A(4) = 4
A(5) = 5 } Присоединенный массив B(I)

B(1) =
Variable not found, 80 : 1

Директива KEYWORD (мир клавиш), задаваемая в виде

KEYWORD *N*

где *N* – целое число от 0 до 4, вводит различные режимы (моды) работы:

мода 0 – обычный режим, мода 1 – некоторые специальные режимы, например набор ключевых слов Бейсика по буквам вместо ввода их разом.

Для обработки данных очень полезна директива сортировки SORT, позволяющая располагать данные в массивах в упорядоченном порядке убывания величин, например

```
 10 DIM A(5)
 20 LET A(1)=10,A(2)=1,A(3)=5,A
(4)=2,A(5)=-3
 30 SORT A()
 50 FOR I=1 TO 5
 60    PRINT "A(";I;")=";A(I)
 70 NEXT I
```

Выполнение этой программы дает следующий результат:

A(1) = 10
A(2) = 5
A(3) = 2
A(4) = 1
A(5) = –3

Директива SORT INVERSE обеспечивает сортировку в обратном порядке – возрастания величин. Так, если в строке 30 вместо SORT A( ) дать директиву SORT INVERSE A( ), то результат выполнения программы будет таким:

A(1) = –3
A(2) = 1
A(3) = 2
A(4) = 5
A(5) = 10

Функция ITEM ( ) (раскладывать по порядку) служит для анализа данных в списке оператора DATA при чтении их оператором READ, анализирует каждый очередной элемент в списке DATA и выдает: 0, если данных нет, 1, если элемент символьного типа, и 2, если элемент числового типа. Функцию ITEM удобно использовать для создания условия окончания цикла считывания данных из списка оператора DATA:

```
10 DO
20    READ X
      PRINT X,"ITEM=";ITEM()
30 LOOP UNTIL ITEM()=1
40 READ A$
   PRINT A$,"ITEM=";ITEM()
50 DATA 1,2,3,4,5,"HELLO"
```

Результат выполнения этой программы следующий:

```
1           ITEM = 2
2           ITEM = 2
3           ITEM = 2
4           ITEM = 2
5           ITEM = 2
HELLO       ITEM = 1
            ITEM = 0
```

Вначале (строки 10–30) из блока DATA (строка 50) считываются в цикле числовые данные. Как только обнаруживается символьный элемент "HELLO", функция ITEM ( ) принимает значение 1, что ведет к выходу из цикла (см. строку 30). В строке 40 переменной A$ присваивается значение слова HELLO. Поскольку больше элементов в списке DATA нет, функция ITEM ( ) принимает значение 0.

Функция

XOR (Целое число 1, Целое число 2)

реализует операцию "исключающее ИЛИ" между двумя целыми числами, вырабатывая 0, если числа одинаковы, и 1, если они различны. Сравнение чисел идет по их битам. Например:

```
PRINT XOR(0, 0)     дает    0,
PRINT XOR(5, 5)     дает    0,
PRINT XOR(5, 1)     дает    1.
```

Многие дополнительные функции могут работать с символьными переменными и константами (эти возможности обсуждаются в § 4.7).

Кроме того, в описываемых версиях Бейсика существенно расширены функции других команд. Некоторые из них отмечены ниже.

Команда редактирования EDIT может использоваться для редактирования численных значений переменных. Для этого она вводится в виде

EDIT ; Имя переменной

Например, введем LET X = 12345678 и PRINT X. Получим вывод числа X: 12345678. Теперь зададим EDIT ; X. Значение X появится в служебной строке редактирования. Используя клавиши перемещения курсора

← и стирания DELETE, устраним цифры 456. Теперь введя команды ENTER PRINT X, получим X = 12378.

Команда стирания DELETE может использоваться для уничтожения элементов массивов. При этом автоматически уменьшается размерность массива. Это иллюстрирует следующая программа:

```
10 DIM a(3)
20 LET a(1)=1,a(2)=2,a(3)=3
30 PRINT a(1),a(2),a(3)
40 DELETE a(2)
50 PRINT "FUNCTION DELETE:"
60 PRINT A(1),A(2),A(3)
```

В строках 10–30 задан массив из трех элементов: a (1) = 1, a (2) = 2, (3) = 3. В строке 40 элемент a (2) уничтожен командой DELETE a (2). При этом массив сокращается и содержит уже два элемента a (1) = 1 и a (2) = 3 (т. е. a (3) переходит на место (2)).

При исполнении программы получим

```
1                              2
3
FUNCTION DELETE
1                              3
Subscript wrong      , 60 :    1
```

Сообщение внизу означает, что превышена размерность массива в строке 60 (действительно, в ней оставлен вывод трех элементов массива, а их осталось два).

Команда COPY (копирование), вводимая в виде

COPY Имя массива 1 (Номер 1) TO Имя массива 2 (Номер 2)

обеспечивает перенос элемента массива 1 с номером 1 в массив 2 с номером 2. Например, в программе

```
10 DIM A(3)
   DIM B(3)
20 LET A(1)=1,A(2)=2,A(3)=3
30 COPY A(2) TO B(3)
40 PRINT a(1),a(2),a(3)
50 PRINT "ARRAY B(3)"
60 PRINT B(1),B(2),B(3)
```

элемент массива A (2) заносится в элемент B (3) массива B. Вывод результата

```
1                 2
3
ARRAY B(3)
0                 0
2
```

Описание выше функции и операторы существенно расширяют возможности программирования сложных расчетных задач.

Глава 4

ПРОГРАММИРОВАНИЕ ОПЕРАЦИЙ С СИМВОЛЬНЫМИ ПЕРЕМЕННЫМИ

4.1. СИМВОЛЬНЫЕ КОНСТАНТЫ И ПЕРЕМЕННЫЕ

Символьными (текстовыми или строчными) *константами* называют группу символов (включая пробелы), заключенную в кавычки: "Старт", "Финиш", "АВ", "123", "1Е28". Их можно рассматривать как часть текста. Группа знаков образует строку, в строке от 6–10 знаков у простейших карманных ПЭВМ до десятков и сотен у большинства ПЭВМ.

Под значением символьной константы подразумевается не число, а начертание всех входящих в нее знаков, даже если они цифры. Поэтому попытки выполнить с символьной константой обычные арифметические операции (например, вычислить 2*"123") воспринимаются ПЭВМ как некорректные и сопровождаются выдачей сообщения об ошибке.

Символьными переменными называют переменные, принимающие значения символьных констант и символьных выражений. Такие переменные обозначаются так же, как и обычные переменные, но после них ставится знак $ или ¤, например: A$ или A¤, X1$ или X1¤, M$(2) или M¤(2), P$(3,5) или P¤(3,5), знак $ или ¤ присваивает переменной статус символьной. Далее будет использоваться знак $.

Символьные константы и переменные характеризуются длиной – общим количеством знаков в них, включая пробелы. С помощью символьных констант и переменных можно формировать тексты комментариев, справок, пояснений и т.д.

Принятые для числовых констант и переменных операции не подходят для символьных констант и переменных, бессмысленными являются операции вычитания, умножения и деления. Например, операция сложения означает просто соединение символов:

"СТАРТ" + "ФИНИШ" = "СТАРТФИНИШ"

Особый смысл имеют и операции сравнения, кроме операции равенства, означающей полное совпадение всех знаков. Например, запись A$ = "СТАРТ" означает, что значение символьной переменной A$ есть слово СТАРТ, то же, что и у константы "СТАРТ".

Каждый символ символьной константы или переменной имеет код в виде числа, который заносится в соответствующую адресуемую ячейку ОЗУ. ПЭВМ выполняет операции над кодами, но при выводе информации на экран дисплея или при ее распечатке принтером последовательности кодов преобразуются в последовательность символов.

Ряд функций символьных переменных обеспечивает вычисление их длины, выбор из строки определенного числа знаков, преобразования символов цифр в их числовые эквиваленты и т.д. Символьные переменные позволяют применять ПЭВМ для хранения, обработки и редактирования текстовой информации, одной из наиболее массовых.

4.2. ВВОД И ВЫВОД СИМВОЛЬНЫХ ПЕРЕМЕННЫХ И ОРГАНИЗАЦИЯ ИХ МАССИВОВ (ОПЕРАТОРЫ INPUT, LET, INPUT LINE, DATA–READ–RESTORE, PRINT, TAB, AT)

Ввод символьной переменной. Осуществляется с пульта ПЭВМ оператором INPUT:

[НС] INPUT ⟨Комментарий⟩ P α$ [P ⟨Комментарий⟩ P β$ . . .]

Здесь α, β – имена переменных (A, B, X1 и т.д.); P – разделитель; запятая, точка с запятой, апостроф. Оператором INPUT можно задавать символьное значение одной или нескольких переменных. Например, выполнение строки

```
1Ø Ø INPUT "БУДЕТЕ СЧИТАТЬ ДАЛЬШЕ – ДА ИЛИ НЕТ ?"; AS
```

приведет к выдаче комментария

БУДЕТЕ СЧИТАТЬ ДАЛЬШЕ – ДА ИЛИ НЕТ ? [" "]

Если пользователь введет слово ДА или НЕТ, оно становится символьным значением переменной A$.

Некоторые ПЭВМ после комментария выводят на индикацию кавычки. Они являются признаком того, что запрашивается символьное, а не численное значение. Однако у многих ПЭВМ кавычки не индицируются.

Разделитель в виде точки с запятой ведет к появлению зоны вводимых символов сразу после комментария. Например, строка

```
1Ø INPUT "ВРЕМЯ СТАРТА"; AS; "ВРЕМЯ ФИНИША"; BS
```

при вводе времени старта 9.00 и времени финиша 10.35 будет индицировать сообщение:

ВРЕМЯ СТАРТА "9.ØØ" ВРЕМЯ ФИНИША "1Ø.35"

Разделитель в виде запятой переносит начало второго комментария в другую позицию строки. Так, строка

```
1Ø INPUT "ВРЕМЯ СТАРТА", AS' "ВРЕМЯ ФИНИША"; BS
```

при вводе будет создавать диалог вида

ВРЕМЯ СТАРТА "9.ØØ"
ВРЕМЯ ФИНИША "1Ø.35"

Наконец, разделитель в виде апострофа ведет к переводу сообщения на новую строку. Строка

```
1Ø INPUT "ВРЕМЯ СТАРТА", AS, "ВРЕМЯ ФИНИША", BS
```

при вводе дает такой диалог:

ВРЕМЯ СТАРТА "9.ØØ"
ВРЕМЯ ФИНИША "1Ø.35"

Для ввода длинных строк с любыми символами используются также операторы

[НС] LINPUT α$ или [НС] INPUT LINE α$

В этих операторах вывод комментариев не предусмотрен. Кроме того, при их использовании не ограничена длина строки. Не следует путать ее с длиной строки дисплея или принтера. Строка символьной переменной может содержать много строк (или даже страниц) информации, выводимой на индикацию или печать.

Присвоение символьной переменной значения символьной константы выполняется с помощью команд вида

```
[НС]   LET α$ = "Символьная константа"
```

или

```
[НС]   α$ = "Символьная константа"
```

Наличие или отсутствие слова LET зависит от применяемой версии Бейсика, например

```
1Ø  LET A$="СТАРТ"
```

или

```
1Ø  A$="СТАРТ"
```

– переменная A$ принимает значение слова СТАРТ.

Ряду символьных переменных можно задавать значения констант с помощью операторов DATA, READ и RESTORE. Более того, допустимо делать это для числовых и символьных переменных одновременно, строго соблюдая их последовательность в строках операторов DATA и READ. Например, во фрагменте программы

```
1Ø  DATA 1, 2, 3, "START", 4
2Ø  READ A, B, C
3Ø  READ D$, E
```

переменным A, B, C и E присваиваются значения 1, 2, 3 и 4, а символьной переменной D$ – значение слова START.

Символьные переменные могут быть индексированными, т.е. переменными одно- и двухмерных массивов. Обычно одна такая переменная может принимать значение только одного символа. Перед использованием подобных переменных нужно зарезервировать память с помощью оператора DIM. Например, выполнение следующей программы:

```
1Ø  DIM X$(5): DATA "S", "T", "A", "R", "T"
2Ø  FOR I=1 TO 5
3Ø  READ X$(I): NEXT I
```

формирует массив из пяти символьных переменных: X$ (1) = "S", X$ (2) = = "T", X$ (3) = "A", X$ (4) = "R" и X$ (5) = "T".

Как и для числовых переменных, для возврата стека символьных данных можно использовать оператор RESTORE, позволяющий присваивать символьные значения разным переменным.

Двухмерные массивы символьных переменных можно понимать как набор слов, расположенных столбцом и имеющих одинаковое число символов. Например, слова

СТАРТ
ФИНИШ

можно представить в виде двухмерного массива

A$(1,1) = "С" A$(1,2) = "Т" A$(1,3) = "А" A$(1,4) = "Р" A$(1,5) = "Т"
A$(2,1) = "Ф" A$(2,2) = "И" A$(2,3) = "Н" A$(2,4) = "И" A$(2,5) = "Ш"

Перед его вводом надо задать директиву DIM(2,5). Отметим, что переменная A$(1) = "СТАРТ" есть просто объединение пяти переменных двухмерного массива A$(1,1), . . . , A$(1,5).

Каждый символ массива занимает в ОЗУ одну ячейку (1 байт). Следовательно, ПЭВМ с ОЗУ пользователя емкостью 64 Кбайт способна вместить примерно 1,5 печатных листа (1 п.л. = 40 000 знаков). Чтобы вместить в ПЭВМ данную книгу нужна память емкостью около 1 Мбайт. Большинство ПЭВМ не имеют ОЗУ столь большой емкости. Поэтому текстовая информация обычно хранится на магнитных дисках или магнитофонных кассетах. Один-три гибких магнитных диска или кассета на 90 мин записи вмещают около 1 Мбайт информации.

Вывод символьных констант и переменных. Для вывода символьных констант и переменных служит оператор PRINT со списком констант и переменных:

[НС] PRINT [Список констант и переменных с разделителями]

В качестве разделителей используются точка с запятой, запятая и апостроф. Действие разделителей аналогично описанному выше для оператора INPUT. Могут выводиться константы различного типа, например при выполнении строки

```
2Ø  PRINT "ВРЕМЯ СТАРТА", 9.3Ø' "ВРЕМЯ ФИНИША", 1Ø.45
```

выводится на индикацию следующее сообщение:

```
ВРЕМЯ СТАРТА                9.3Ø
ВРЕМЯ ФИНИША                1Ø.45
```

Если A$ = "ВРЕМЯ СТАРТА" и B$ = "ВРЕМЯ ФИНИША", то аналогичный результат будет получен при выполнении строки

```
2Ø  PRINT A$, 9.3Ø' B$, 1Ø.45
```

или строки

```
3Ø  PRINT A$, "9.3Ø" B$, "1Ø.45"
```

В последнем случае числа "9.3Ø" и "1Ø.45" используются как символьные константы. У некоторых ПЭВМ значения символьных констант выводятся на индикацию в кавычках.

В состав оператора PRINT могут вводиться модификаторы TAB и AT, действие которых аналогично описанному для вещественных констант и переменных. Например, выполнение строки

```
3Ø  PRINT A$; "9.3Ø"; TAB 15 ; B; "1Ø.45"
```

дает печать сообщений

ВРЕМЯ СТАРТА 9.3Ø ВРЕМЯ ФИНИША 1Ø.45

причем второе сообщение всегда будет начинаться с позиции 15 (задается модификатором TAB 15).

4.3. ФУНКЦИИ СИМВОЛЬНЫХ ПЕРЕМЕННЫХ (LEN, STR$, VAL, VAL$, LEFT$, RIGHT$, MID$, TO И ДР.)

Для выполнения различных операций с текстами служат функции символьных переменных. Ниже перечислены основные из них.

Вычисление длины символьной константы или переменной. Эта функция LEN (от слова lenght – длина) создает число, равное числу знаков ее аргумента (включая пробелы), в виде символьной константы или переменной, записывается в виде

[НС] LEN [(]Аргумент[)]

Например, выполнение команды PRINT LEN ("START") или PRINT LEN "START" выводит на индикацию или печать число 5 – столько знаков в слове START.

У некоторых ПЭВМ скобки обязательны. Есть ПЭВМ (например, FX-702P), у которых аргументом функции LEN может быть только переменная, но не константа.

Преобразование числа в значение символьной переменной. Число преобразуется в символьное значение символьной переменной с помощью функции STR$ (от слова strophe – строфа):

$$\text{STR\$}\,[(]\begin{Bmatrix}\text{Константа}\\ \text{Имя вещественной переменной}\\ \text{Арифметическое выражение}\end{Bmatrix}[)]$$

Так, если X=4, то выполнение команд LET A$=STR$(4), LET A$ = = STR$(X) и LET A$=STR$(2*2) придает символьной переменной значение символа 4.

Незначащие нули мантиссы аргумента функции STR$ игнорируются. Например, выполнение команды PRINT STR$(1Ø.25ØØØØØØ) приведет к индикации строки 10.25. При выполнении функции STR$ числа в аргументе нормализуются. Например, при выполнении команды PRINT STR$(123456789ØØØ) будет выведена на индикацию строка 1.2345679E+11. Обратите внимание на потерю цифры 8: она округлена до 9, поскольку последующая за ней цифра 9 не вмещается в формат представления мантиссы.

Напоминаем, что с символьными значениями нельзя выполнять арифметические операции. Например, попытка выполнить команду PRINT 2*STR$(2) будет неудачной: ПЭВМ либо откажется воспринимать такую команду, либо выдаст сообщение об ошибке.

Преобразование символьного значения в числовое. Символьное значение

в виде числа преобразуется в числовое значение с помощью функции VAL (от слова value – значение):

[НС] VAL [(] Аргумент [)]

Аргументом может быть символьная константа (например, "123") или переменная (например, X$). Так, если X$ = "123", то выполнение команды LET Y = VAL ("123") или LET Y = VAL(X$) придает вещественной переменной Y числовое значение 123.

При преобразовании числа нормализуются. Например, при выполнении команды PRINT VAL ("123456789ØØØ") выдается число в виде 1.23456794+ +11 (здесь также из-за округления оказалась потерянной цифра 8 мантиссы). У некоторых ПЭВМ (например, ZX-Spectrum) в составе аргумента могут быть знаки арифметических операций, например выполнение команды PRINT VAL "3*4" дает число 12, а PRINT VAL ("3" + "4") – число 7. Если A$ = "3*4", то PRINT VAL A$ или PRINT VAL (A$) дает число 12. Таким образом, пока числа находятся в кавычках, они ведут себя как числа.

С числами, создаваемыми функцией VAL, можно проводить любые арифметические действия, например PRINT 4*VAL "5" дает число 2Ø, PRINT VAL "3" + VAL "4" число 7, PRINT EXP (VAL "2") e^2 = 7.389Ø561.

Преобразование символьного значения в строку. У некоторых ПЭВМ вывод символьной переменной задается в кавычках. Например, при выполнении строк

```
1Ø  LET A$="START"
2Ø  PRINT A$
```

индицируется сообщение "START".

Функция VAL$ выделяет символы и исключает обрамляющие кавычки. Так, при выполнении строк

```
1Ø  LET A$= """START"""
2Ø  PRINT VAL$ (A$)
```

индицируется сообщение START.

Выделение крайних левых символов строки. Функция LEFT$ обеспечивает выделение *n* левых символов в строке, записывается в виде

$$\text{[НС]}\quad \text{LEFT\$}\ \left(\left\{\begin{matrix}\text{"Константа"}\\ \text{Переменная}\end{matrix}\right\}, n\right)$$

Например, выполнение строк

```
1Ø  LET A$="STARTFINISH"
2Ø  PRINT LEFT$(A$, 5)
```

обеспечивает печать слова START – 5 знаков строки STARTFINISH слева. Если *n* больше числа символов константы или переменной, выделяются все символы.

Выделение крайних правых символов строки. Функция RIGHT$, обес-

печивающая выделение n крайних правых символов в строке, задается в виде

$$[\text{HC}] \quad \text{RIGHT\$}\left(\left\{\begin{matrix}\text{"Константа"}\\ \text{Переменная}\end{matrix}\right\}, n\right)$$

Так, выполнение строк

```
10 LET A$="START FINISH"
20 PRINT RIGHT$(A$, 6)
```

обеспечивает печать слова FINISH – 6 знаков строки STARTFINISH справа. Если n больше числа символов в строке, выделяется вся строка.

Выделение подстроки. Подстрокой называется часть строки. Подстрока характеризуется порядковым номером первого символа и длиной подстроки. Например, в строке

A$ – "ПОНВТСРЧЕТПЯТНСУБ"

подчеркнутая подстрока ПЯТН имеет номер первого символа $n_1 = 11$ и длину $n_2 = 4$. Обычно выделение подстроки обеспечивает функция MID$ (от слова middle – в середине) вида

$$[\text{HC}] \quad \text{MID}(\alpha\$, n_1\ [, n_2])\,,$$

где α – имя переменной. Так, выполнение строки

```
50 LET B$=MID$(A$, 11,4)
```

придает символьной переменной B$ значение "ПЯТН".

Действие текстовых функций иллюстрирует следующая программа:

```
10 A$="STARTFINISH":PRINT"A$=";A$
20 B$=STR$(2*3)
30 PRINT"LEN(A$)=";LEN(A$)
35 N=VAL(B$):PRINT"N=VAL(B$)=";N
40 PRINT"STR$(2*3)=B$=";B$
50 C$=LEFT$(A$,5)
60 PRINT"C$=LEFT$(A$,5)=";C$
70 PRINT"RIGHT$(A$(6)=";RIGHT$(A$,6)
80 PRINT"MID$(A$,6,6)=";MID$(A$,6,6)
```

```
A$=STARTFINISH
LEN(A$)= 11
N=VAL(B$)= 6
STR$(2*3)=B$= 6
C$=LEFT$(A$,5)=START
RIGHT$(A$,6)=FINISH
MID$(A$,6,6)=FINISH
```

У ПЭВМ FX-702P функция MID действует только в отношении особой символьной переменной, длина которой может достигать 30 символов. При этом функция MID записывается в виде

$$[\text{HC}] \quad \text{MID}(n_1\ [, n_2])$$

Если n_2 не указано, то выделяется вся подстрока после символа под номером n_1. Например, если $= "123456789", то MID(4) дает строку 456789, а MID(3.4) – строку 3456.

У ПЭВМ ZX-Spectrum функция MID$ отсутствует, ее заменяет особое выражение

$$\left\{\begin{matrix}\text{"Константа"}\\ \alpha\$\end{matrix}\right\}([n_1]\ \text{TO}\ [n_2])$$

где n_1 – номер первого, а n_2 – номер последнего символа подстроки, заданной символьной константой или переменной $\alpha\$$. Приведенные ниже примеры иллюстрируют выделение подстроки или отдельного индекса:

"abcdefl" (3 TO 6) = "cdef"
"abcdefi" (2 TO) = "bcdefl"
"abcdefl" (TO 4) = "abcd"
"abcdefi" (TO) = "abcdefl"
"abcdefl" (4 TO 4) = "d"
"abcdefl" (1 TO ∅) = " "
"abcdefl" (9 TO 8) = " "
"abcdefi" (6 TO 8) – выводится сообщение об ошибке (3 Subscript wrong), указывающее на то, что индекс 8 превысил предельное значение $n_2 = 7$.

Выделение последовательностей знаков в цикле с оператором вида (1 TO I):

```
10 REM ОПЕРАТОР TO
20 LET F$="АБВГДЕЖЗИК"
30 FOR I=1 TO 10
40 PRINT F$(1 TO I)
50 NEXT I
```

```
А
АБ
АБВ
АБВГ
АБВГД
АБВГДЕ
АБВГДЕЖ
АБВГДЕЖЗ
АБВГДЕЖЗИ
АБВГДЕЖЗИК
```

Наряду с описанными в ряде расширенных версий Бейсика встречаются и другие функции:

TRM(n) – формирует строку без конечных пробелов,

POS$(\alpha\$, \beta\$, n)$ – обеспечивает поиск подстроки $\beta\$$ в строке $\alpha\$$ начиная с позиции n,

SEG$\$(\alpha\$, n_1, n_2)$ – выделяет подстроку из строки $\alpha\$$ начиная с позиции n_1 и кончая n_2,

OCT$(\alpha\$)$ – дает десятичное значение восьмеричного числа-строки,

MKI$\$(\alpha\%)$ – преобразует двоичное число-строку $\alpha\%$ в целое десятичное число,

MKS$\$(\alpha!)$ – преобразует четверичное (4-байтовое) число-строку в десятичное число обычной точности,

MKD$\$(\alpha\#)$ – преобразует восьмеричное число-строку в десятичное число обычной точности,

CI$(\alpha\%)$, CS$(\alpha!)$ и CD$(\alpha\#)$ – преобразует символьные переменные $\alpha\%$ в виде 2, 4 и 8-байтовых чисел в числовые значения их,

SPACE ¤ (n) – создает строку из n пробелов,

STRING ¤ (n_1, n_2) – создает строку из n_1 символов с кодом n_2,

STRING ¤ (n, α ¤) – создает строку из n символов, каждый из которых есть первый символ переменной (или константы) α ¤,

BIN$ (n) – создает символьную цепочку-двоичное представление десятичного целого числа n (например, BIN$ (254) дает "11111110"),

CHAR$($n$) – переводит целые десятичные числа (от 0 до 65535 в двухсимвольную цепочку для их более экономного хранения в ОЗУ (числа при этом занимают 2 байт),

NUMBER($\alpha$$, $\beta$$) – преобразует двухсимвольную цепочку в целое десятичное число (от 0 до 65535),

HEX$ (n) – преобразует целое десятичное число n в шестнадцатеричное число-строку.

Функция

INSTRING(n, $\alpha$$, $\beta$$)

– отыскивает цепочку символов $\beta$$ в цепочке символов $\alpha$$ начиная с позиции n. Если поиск прошел успешно, функция вырабатывает результат в виде числа-номера позиции первого символа $\beta$$ в строке $\alpha$$. Если цепочка $\beta$$ длиннее $\alpha$$ или длина $\alpha$$ или $\beta$$ равна 0, то функция вырабатывает нуль. Нуль вырабатывается и при отрицательном результате поиска. Например, команда

PRINT INSTRING(1, A$, "SMITH")

– дает 3, если A$ = "YESSMITH, и 0, если A$ = "YESMATH".

Существует возможность задания в $\beta$$ знака произвольного невоспринимаемого символа #. В этом случае в цепочке $\alpha$$ может стоять любой знак. Так, команда

PRINT INSTRING(1, A$, "SM#TH")

теперь будет давать 1 для A$ = "SMITH", A$ = "SMATH" и т.д. Символы I и A в этом примере не учитываются при поиске $\beta$$ в $\alpha$$.

Функция

MEMORY$( ) [([$m$] TO [$n$])]

выдает содержимое ОЗУ (ячеек от m до n) как цепочку символов. Таким образом, можно скопировать как значения символьных переменных определенные участки ОЗУ (например, отведенные под фрагменты изображения или программы). Так, при выполнении команды

LET P$ = MEMORY$( ) (25000 TO 25672)

переменная P$ хранит содержимое ОЗУ с ячейки с адресом 25000 до ячейки с адресом 25672.

У версий Бейсика с функцией MEMORY$ используется директива, переносящая содержание этой функции в ОЗУ с установкой начала на любой адрес, вида

POKE n, $\alpha$$

где n – начальный адрес; $\alpha\$$ – цепочка символов, хранящая содержимое ОЗУ. Например, по команде

```
POKE 43ØØØ, MEMORY$ ( ) (25ØØØ TO 25672)
```

или

```
POKE 43ØØØ, A$
```

загружается содержимое ячеек с 25000 до 25672 в другие ячейки ОЗУ теперь с адреса 43000. Так с помощью функций MEMORY$ и директивы POKE n, $\alpha\$$ можно создавать в ОЗУ копии различных участков, перемещать содержимое ОЗУ с одного места на другие и т.д.

В ряде версий Бейсика предусматривается возможность вывода даты и текущего времени (или времени с начального момента включения часов). Функция DATE ¤ (или DATE$) создает символьное значение текущей даты в виде

ЧЧ – МММ – ГГ,

где ЧЧ – две цифры числа, три буквы месяца и ГГ – две буквы года (например, 12 – ЯНВ – 87 означает 12 января 1987 г). Функция TIME$ создает символьное значение текущего времени в виде

ЧЧ – ММ – СС,

где ЧЧ – две цифры часов; ММ – две цифры минут; СС – две цифры секунд (например, ØØ – 12 – 36, т.е. 0 часов 12 минут и 36 секунд). Функция TIME$ аналогична функции CLK ¤. Разделителем может быть знак : вместо – .

4.4. ОПЕРАЦИИ С СИМВОЛЬНЫМИ ПЕРЕМЕННЫМИ И КОНСТАНТАМИ

Рассмотрим некоторые дополнительные операции, которые можно выполнять с символьными переменными и константами.

Соединение символьных строк. Строки могут складываться с помощью знака +, например

```
LET A$ = "START" + "FINISH" + "END"
```

Эта операция означает соединение символьных констант – строк:

```
A$ = "STARTFINISH END"
```

Если B$ = "START", C$ = "FINISH" и D$ = "END", то выполним операцию A$ = B$ + C$ + D$, получим так же результат.

Операция соединения обычно может входить в функции, аргументом которых являются символьные переменные. Например, выполнение программы

```
1Ø  LET AS="START"
2Ø  LET BS="FINISH"
3Ø  LET CS="END"
4Ø  PRINT LEN(AS+BS+CS)
```

даст число 14 – общее число знаков в строке STARTFINISHED. Другой пример (для ПЭВМ ZX-Spectrum):

"abc"+"def" (1 TO 2)="abcde"

Здесь операция выделения двух первых символов (1 TO 2) относится к символьной константе "def". А в данном случае

("abc"+"def") (1 TO 2)="ab"

та же операция относится к сумме (соединению) констант "abc" и "def", т.е. к результирующей константе "abcde", полученной в результате соединения констант "abc" и "def", приведенных в скобках.

Сравнение символьных строк. Символьные строки (константы и переменные) могут подвергаться операциям сравнения. Примеры таких операций:

Операция	Пример	Комментарий
=	A$=B$	Переменная A$ равна переменной B$
<	A$<B$	Переменная A$ меньше переменной B$
>	A$>B$	Переменная A$ больше переменной B$
>=, =>	A$>=B$	Переменная A$ больше или равна переменной B$
<=, =<	A$=B$	Переменная A$ меньше или равна переменной B$
<>, ><	A$><B$	Переменная A$ не равна переменной B$

Равенство понимается в смысле полного совпадения строк. Производится сравнение не самих символьных переменных, а кодов их символов, оно может выполняться различным способом. У некоторых ПЭВМ сравнение производится только по кодам первых символов каждой строки. При расположении латинских букв в алфавитном порядке коды ASCII идут в возрастающем порядке. Поэтому "A"<"B", "B"<"C" и т.д. В кодах КОИ-7 и КОИ-8 порядок кодов русских букв не совпадает с их алфавитным порядком. Поэтому, применяя строки с русскими буквами в операциях сравнения, нужно учитывать их коды, а не алфавитное расположение. В некоторых ПЭВМ выполняется сравнение многосимвольных строк по сумме кодов строк, которые вычисляются для символов слева направо, короткая строка дополняется пробелами.

Операторы IF, THEN и их модификаторы. Совместно с символьными строками и операциями сравнения (ОС) применяются операторы IF, THEN и их модификаторы (М):

$$[\text{НС}]\quad \text{IF STR1}\ \begin{Bmatrix}\text{OC1}\\ \text{M1}\end{Bmatrix}\ \text{STR2}\ \begin{Bmatrix}\text{OC2}\\ \text{M2}\end{Bmatrix}\ \ldots\ \text{THEN}\quad \text{Инструкции}$$

Например, при выполнении строк программы

```
1Ø  IF A$ AND B$ =C$ THEN PRINT "C$=";  C$:STOP
11Ø  PRINT "END"
```

индицируется символьное значение C$, если A$ и B$ равны C$, иначе будет выдана строка END. Применяются модификаторы AND (и), NOT (нет), OR (или) и др.

Условные переходы. Символьные строки могут входить в состав условных переходов

$$[\text{HC}]\quad \text{IF STR1}\left\{\begin{matrix}\text{OC1}\\ \text{M1}\end{matrix}\right\}\ \text{STR2}\left\{\begin{matrix}\text{OC2}\\ \text{M2}\end{matrix}\right\}\ \ldots\ \text{THEN [GOTO] HC1}$$

где HC1 – номер строки, по которому происходит переход. Так, выполнение фрагмента программы

```
1ØØ  IF A$ = C$  AND B$ =C$ THEN GOTO 12Ø
11Ø  PRINT "E"  " : STOP
12Ø  PRINT "C$ ="; C$
```

если **A$** и **B$** равны **C$**, дает результат, аналогичный приведенному выше, но за счет условного перехода к строке 120.

Операции сравнения в арифметических выражениях. В некоторых версиях Бейсика (например, ПЭВМ ZX-Spectrum) операции сравнения любого вида констант и переменных могут в качестве функций входить в состав арифметических выражений. Если операция сравнения истинна, то выражение в скобках с такой операцией принимает значение 1, если ложно, то 0. Например, если **A$** = "A" и **B$** = "B", то (A$ = B$) = 0, а (A$ <> B$) = 1, так как в первом случае условие (=) не выполняется ("A" ≠ "B"), а во втором выполняется ("A" ≠ "B" или "A" < > "B"). Аналогично PRINT (X = 2) дает значение 0 при любом X, кроме 2, в последнем случае получаем 1. В подобных выражениях могут использоваться модификаторы AND, OR и NOT. Например, в программе

```
1Ø  LET A$ ="СТАРТ": LET B$ ="СТАРТ"
2Ø  INPUT X$
3Ø  LET Y =(X$ =A$ AND X$ =B$)
4Ø  IF Y = 1 THEN PRINT "СТАРТ"
5Ø  PRINT "НЕТ СТАРТА": GOTO 2Ø
```

Y = 1 только в том случае, если введено символьное значение X$ = "СТАРТ" (см. строку 30). В этом случае (строка 40) индицируется сообщение СТАРТ. Если ввести другое значение **X$**, будет индицироваться сообщение НЕТ СТАРТА.

В расширенных версиях Бейсика имеется ряд дополнительных операторов для работы с символьными переменными. Ниже описаны эти операторы (версия Beta-BASIC = 3.0).

Оператор ALTER (замена), записываемый в виде

ALTER "Слово 1" TO "Слово 2"

обеспечивает замену слова 1 на слово 2 по всей программе. Введем, к примеру, строку

```
1Ø  LET A$ ="СТАРТ"
```

и далее в непосредственном режиме исполним команду

ALTER "СТАРТ" TO "ФИНИШ"

Если теперь командой LIST 1Ø вывести строку 10, она индицируется в виде

```
1Ø  LET A$="ФИНИШ"
```

Аналогично, оператор

ALTER Имя 1 TO Имя 2

обеспечивает по всей программе замену имени переменной 1 именем переменной 2. Например, дав команду

ALTER A$ TO B$

получим строку 10 в виде

```
1Ø  LET B$="ФИНИШ"
```

Команду ALTER можно использовать и для стирания символьной переменной. Так, команда

ALTER "ФИНИШ" TO " "

меняет строку 10 на

```
1Ø  LET B$=" "
```

Теперь переменная B$ становится пустой (если между кавычками нет пробела).

Команда копирования COPY в виде

COPY Имя 1 TO Имя 2

заносит копию символьного значения переменной с именем 1 в символьное значение переменной с именем 2, сохраняя значение последней. Так, если A$ = "СТАРТ" и B$ = "ФИНИШ", то команда COPY A$ TO B$ меняет значение переменной B$ на "СТАРТФИНИШ". При этом значение A$ = = "СТАРТ" не меняется.

Для включения в переменную с именем 2 части знаков (от *m* до *n*) переменной с именем 1 можно использовать и такую форму оператора копирования:

COPY Имя 1 [([*m*] TO [*n*])] TO Имя 2

С символьными переменными можно применять и оператор первого присваивания DEFAULT, например

DEFAULT A$ = "СТАРТ"

В этом случае, если ранее переменной A$ никаких символьных значений не присваивалось, исполнение данной команды задаст переменной значение слова СТАРТ. Если присваивания были, то данная команда игнорируется.

Команда стирания

DELETE Имя [([*m*] TO [*n*])]

уничтожает в символьной переменной с указанным именем символы, начиная с позиции *m* и кончая *n*. Так, если A$ = "СТАРТ ФИНИШ", то команда

DELETE A$ (6 TO 11)

стирает пробел и слово ФИНИШ (A$ = "СТАРТ").

Директива редактирования символьной переменной

EDIT Имя

выводит значение переменной с указанным именем на редактирование в служебную строку. При этом текст (значение) этой переменной можно редактировать. По окончании редактирования переменная приобретает символьное значение, скорректированное редактором. Такая возможность редактирования очень удобна при построении банков текстовых данных, создании записных книжек и текстовых редакторов.

Оператор

GET Имя

присваивает переменной с указанным именем символ нажимаемой клавиши. Так, всего одна строка программы

```
1Ø  GET X$: PRINT X$; : GOTO 1Ø
```

при пуске превращает ПЭВМ в аналог пишущей машинки. Можно вводить тексты, нажимая соответствующие клавиши. При этом действует система редактирования текста: клавиши забоя и пробела, перемещения маркера вверх и вниз, вправо и влево.

Оператор

JOIN Имя 1 [([*m*] TO [*n*])] TO Имя 2

подключает символьное значение переменной 1 (начиная с позиции *m* и кончая *n*) к символьному значению переменной с именем 2. Так, если A$ = = "СТАРТ ФИНИШ", а B$ = "СТАРТ", то команда

JOIN A$ (1 TO 6) TO B$

делает значения переменных следующими:

A$ = "ФИНИШ" и B$ = "СТАРТ ФИНИШ"

Таким образом, слово СТАРТ с пробелом из символьного значения A$ исчезает и переходит к переменной B$. Отметим, что команда JOIN X$ TO Y$ делает символьное значение переменной X$ значением переменной Y$, при этом значение X$ исчезает – переменная X$ теряет определение.

Применительно к символьным переменным действует и оператор сортировки SORT. Так, по команде

SORT Имя [([*m*] TO [*n*])]

сортируются символьные элементы (от *m* до *n*) массива с указанным именем в порядке их убывания. Сортировка ведется путем побитового сравнения кодов. Команда

SORT Имя

сортирует все элементы массива. Например, SORT A$ – сортируются элементы массива A$, SORT B$(1Ø TO 2Ø) – элементы массива B$ от 10-го по 20-й. Добавление после слова SORT слова INVERSE меняет порядок сортировки – он идет в направлении роста кодов.

4.5. ПРИМЕНЕНИЕ СИМВОЛЬНЫХ ПЕРЕМЕННЫХ ДЛЯ ОРГАНИЗАЦИИ ДИАЛОГА С ПЭВМ

Метод словаря. Одним из методов сокращения длины программ является метод словаря – введение в программу ряда символьных переменных со значениями строк, соответствующими нужным и часто применяемым словам. Например, ввод ряда значений переменных A, B, . . . , Z можно выполнить обычным путем:

```
1Ø  PRINT "ВВЕДИТЕ ЗНАЧЕНИЕ A=" ; : INPUT A :PRINT A
. . . . . .
26Ø PRINT "ВВЕДИТЕ ЗНАЧЕНИЕ Z ="; : INPUT Z: PRINT Z
```

В этом случае придется 26 раз набирать слова ВВЕДИТЕ ЗНАЧЕНИЕ. Более рациональной будет программа

```
5     LET A$ = "ВВЕДИТЕ ЗНАЧЕНИЕ"
1Ø    PRINT A$; "="; : INPUT A: PRINT A
. . . . . .
26Ø   PRINT A$; "="; :INPUT Z: PRINT Z
```

Здесь в строке 5 переменной A$ придается символьное значение "ВВЕДИТЕ ЗНАЧЕНИЕ", которое при работе программы выводится 26 раз. Поэтому эта программа получается более компактной.

Метод вопросов и ответов. При использовании символьных переменных этот метод является наиболее удачной формой организации диалога с ПЭВМ. Например, при выполнении следующей программы:

```
1Ø  INPUT "ВВЕДИТЕ X=", X
2Ø  INPUT "УКАЖИТЕ, ЧТО ВЫЧИСЛЯЕТСЯ: SQR, EXP, LOG?," A$
3Ø  IF A$="SQR" THEN PRINT SQR(X): GOTO 1Ø
4Ø  IF A$="EXP" THEN PRINT EXP(X): GOTO 1Ø
5Ø  IF A$="LOG" THEN PRINT LOG(X): GOTO 1Ø
6Ø  GOTO 2Ø
```

появляется сообщение

ВВЕДИТЕ X=

Ввод X приводит к новому сообщению-вопросу:

УКАЖИТЕ, ЧТО ВЫЧИСЛЯЕТСЯ: SQR, EXP, LOG?

Допустим, нужно выполнить вычисление e^x = EXP(X). Тогда надо ввести ответ EXP. После этого переменной A$ присваивается символьное значение "EXP" и будет выполнена строка 40 – вычисление EXP(X) с выдачей на индикацию. Любой ответ кроме SQR, EXP и LOG игнорируется, что делает безусловный переход в строке 60.

Этот метод не обязательно требует применения символьных переменных. Более того, расширение диалога делает их применение просто неудобным. Например, пользователь решил указать операции более подробно:

1 ВЫЧИСЛЕНИЕ КВАДРАТНОГО КОРНЯ
2 ВЫЧИСЛЕНИЕ ЭКСПОНЕНТЫ
3 ВЫЧИСЛЕНИЕ НАТУРАЛЬНОГО ЛОГАРИФМА

Очевидно, что ввод таких длинных ответов крайне неудобен. Лучше воспользоваться методом задания кодов, как в следующей программе:

```
10 INPUT "ВВЕДИТЕ X= ", X
20 PRINT "ВЫЧИСЛЕНИЕ КВАДРАТНОГО КОРНЯ – КОД 1"
30 PRINT "ВЫЧИСЛЕНИЕ ЭКСПОНЕНТЫ – КОД 2"
40 PRINT "ВЫЧИСЛЕНИЕ НАТУРАЛЬНОГО ЛОГАРИФМА – КОД 3"
50 PRINT "ВВЕДИТЕ КОД"; :INPUT C: PRINT C
60 IF C=1 THEN PRINT SQR(X): GOTO 10
70 IF C=2 THEN PRINT EXP(X): GOTO 10
80 IF C=3 THEN PRINT LOG(X): GOTO 10
90 GOTO 20
```

Этот пример приведен лишь как альтернатива к применению символьных переменных. Если число операций, выполняемых программой, велико (например, вычисляется много различных функций), то числовое кодирование становится менее удобным, чем символьное. Разумеется, метод вопросов и ответов легко распространить и на программы, ориентированные на задачи не вычислительного характера, например разгадка кроссвордов и т.д.

Метод соответствия. Одной символьной переменной можно поставить в прямое соответствие другую (или группу других переменных). Этот метод очень удобен при составлении словарей для перевода из одного языка на другой, записных книжек, телефонных справочников и т.д. В качестве примера рассмотрим простейшую программу-словарь для перевода с английского языка на русский:

```
100 INPUT X$
101 IF X$="START" THEN PRINT "СТАРТ": GOTO 100
102 IF X$="FINISH" THEN PRINT "ФИНИШ": GOTO 100
103 IF X$="ABYSS" THEN PRINT "БЕЗДНА, ПРОПАСТЬ,
          ПУЧИНА": GOTO 100
```

Работа этой программы столь очевидна, что не требует особых пояснений. Она удобна для небольших словарей или "записных книжек". Если ключевых слов несколько сотен и более, ответ будет выдаваться с заметной задержкой. Две рекомендации помогут создавать более эффективные словари: 1) располагайте наиболее часто встречающиеся слова в начале программы; 2) разбейте слова на группы в порядке алфавита. Тогда нетрудно организовать вначале поиск группы слов с одинаковой первой буквой, а затем в пределах группы нужное слово.

Глава 5
ПРОГРАММИРОВАНИЕ ГРАФИКИ И ЗВУКОВЫХ ЭФФЕКТОВ

5.1. ЗАДАНИЕ ГРАФИЧЕСКИХ И ТЕКСТОВЫХ СТРАНИЦ (ДИРЕКТИВЫ GR, MGR, TEXT)

Страницей называется один кадр изображения, получаемого на экране дисплея. Страницы могут быть трех видов: графические, текстовые и смешанные – содержащие и графику, и различные тексты.

Информация, отображаемая на странице, хранится в оперативном запоминающем устройстве (ОЗУ) ПЭВМ. Каждая точка черно-белого изображения задается своим адресом, хранящимся в ОЗУ. В специальной ячейке ОЗУ хранятся также коды – логический 0, если точка погашена, и 1, если она выводится на индикацию. В общем случае необходимо хранить дополнительную информацию о цвете и яркости точки, режиме ее мигания и т.д. В результате вся эта информация будет занимать уже несколько ячеек ОЗУ и потребуется большая память.

Современные ЭЛТ позволяют формировать изображения, содержащие до 1000X1000 точек. Однако в этом случае затраты памяти могут оказаться чрезмерными, особенно для ПЭВМ с умеренной стоимостью.

Разрешающей способностью (разрешением) *графики* ПЭВМ называют число элементарных графических элементов (точек) по горизонтали и вертикали экрана дисплея.

Типовые затраты памяти (емкости ОЗУ) для графики с заданным разрешением:

Число цветов	Число элементов		Емкость ОЗУ, Кбайт
	графических	текстовых	
2	640X256	32X80	20
4	320X256	32X40	20
16	256X160	32X20	20
8	256X176	32X22	16
2	–	25X80	16
2	320X256	32X40	10
4	256X160	32X20	10
16(2)	40X24 (80X72)	24X40	2
2	–	25X40	1

Поскольку емкость ОЗУ ПЭВМ ограничена (16–64 Кбайт), то графические изображения формируются из укрупненных элементарных элементов – пикселей. *Пиксель* – это обычно прямоугольник или квадрат, содержащий несколько точек, создаваемых ЭЛТ дисплея ПЭВМ.

Связь разрешающей способности ПЭВМ с числом точек в пикселе характеризуется следующими типовыми данными:

Разрешающая способность графики	Число точек в пикселе	Число пикселей в кадре
Низкая	4×4 3×4	64×64 80×72
Средняя	2×2	128×128
Высокая	1×1 1×1	256×256 512×512
Сверхвысокая	1×1	1024×1024

Задание графической страницы. У некоторых ПЭВМ (Агат, Apple-II) переход в режим графики требует специального задания графических страниц. Для этого служат операторы:

GR = α ($2 \leqslant \alpha \leqslant 31$) – задание графической страницы низкого разрешения (64×64),

MGR = α ($1 \leqslant \alpha \leqslant 7$) – задание графической страницы среднего разрешения (128×128),

HGR = α ($1 \leqslant \alpha \leqslant 7$) – задание графической страницы высокого разрешения (256×256).

Задание текстовой страницы. У большинства ПЭВМ текстовая страница особо не задается. Она характеризуется числом знакомест $m \times n$, где m – число строк и n – число знаков в строке (число столбцов). Текстовая страница размера 22×32 показана на рис. 5.1. Однако у ПЭВМ с раздельным формированием графических и текстовых страниц последние задаются специальными операторами, например

TEXT = α ($2 \leqslant \alpha \leqslant 31$)

где α – константа, переменная или арифметическое выражение. Такие ПЭВМ позволяют быстро менять страницы с различными текстами и графиками.

Задание графических и текстовых страниц означает изъятие памяти под них и уменьшение оставшейся емкости ОЗУ, доступной пользователю для других нужд. Поэтому значения α ограничены.

Задание смешанных страниц. Большинство современных ПЭВМ позволяет формировать смешанные страницы с любым видом информации на них. Часто задается одна смешанная страница и под нее отводится специальная область ОЗУ (ОЗУ экрана или дисплея). Эта область является адресуемой – пользователь, указав ее адрес, заносит в нужную ячейку информацию о знаке, отображаемом в заданном знакоместе, или о пикселе графи-

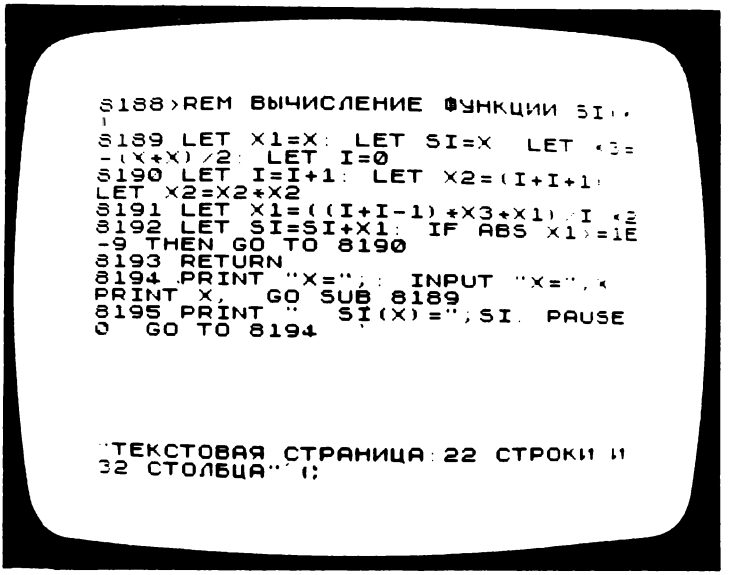


Рис. 5.1

ческого изображения. В ней же хранится информация о различных световых и цветовых эффектах (атрибуты графики).

Графические элементы сложных изображений, формируемые в матрице (обычно 8X8 пикселей) знакоместа называют *графемами*. Вывод графем производится точно так же, как и любых алфавитно-цифровых знаков. Соединяя различные графемы друг с другом, можно создавать сложные, в том числе динамические, изображения с высокой разрешающей способностью при малых затратах емкости ОЗУ (сами графемы хранятся в ПЗУ).

Примером эффективного использования графем является домашняя ПЭВМ Aquarius фирмы Mattel Electronic (Гонконг), в ПЗУ которой хранится около 200 графем. В их числе геометрические фигуры, части фигуры человечков, очертания самолетов, ракет, лазерных установок и т. д. В матрице экрана 24X40 помещается 960 графем. Поскольку каждая имеет 8X8 элементов, то разрешающая способность рисунков достигает 192X320 элементов. Разумеется, при этом набор рисунков ограничен имеющимися графемами. Однако при большом их числе этот набор весьма обширен.

У ряда ПЭВМ предусматривается несколько режимов (мод) отображения информации на экране дисплея. Моды вводятся с помощью операторов MODE *N* или SCREEN *N*, где *N* – номер моды. Так, у ПЭВМ MSX японской фирмы "Ямаха" оператор SCREEN *N* вводит 4 моды:

N = 0: отображение стандартных текстов в виде 24 строк по 40 знаков в каждой строке с одним цветом всех знаков и знакомест;

$N = 1$: отображение текстов в виде 32 строк по 40 знаков в каждой строке с раздельной установкой цветов каждого знака и каждого знакоместа, в этой моде графика также не выводится;

$N = 2$: графика высокого разрешения (256X192 элементов) с установкой одного цвета для каждого элемента (пикселя) и общего фона экрана; точка (0, 0) расположена в левом верхнем углу экрана;

$N = 3$: графика низкого разрешения (64X48 элементов), но с установкой любого из 16 цветов для каждого элемента и его знакоместа (включая изображения графем).

Во многих ПЭВМ (например, MSX или ZX-Spectrum) предусмотрена возможность создания большого числа графем пользователем. Это позволяет создавать сложные обучающие и игровые программы, машинные фильмы с движущимися объектами, составленными из графем.

5.2. ЗАДАНИЕ ЦВЕТА (ОПЕРАТОРЫ COLOR, INK, PAPER, BORDER, BRIGHT, FLASH, OVER, NORMAL, INVERSE)

В общем случае формируемое на экране дисплея изображение содержит рабочее окно (собственно страницу) и ее окаймление-бордюр.

Цветная графика возможна только в том случае, если в ПЭВМ имеется адаптер цветной графики и цветной телевизор работает в той же системе передачи цветного изображения, что и адаптер ПЭВМ. В противном случае цвета не воспринимаются, но будут выглядеть как градации яркости (это само по себе также полезно, ибо расширяет возможности создания черно-белых изображений, делая их полутоновыми). На рис. 5.2 показано, как выглядит изображение, снятое с экрана черно-белого телевизора системы SECAM, подключенного к адаптеру ПЭВМ с системой PAL. Тот же результат будет, если ПЭВМ с адаптером системы SECAM (принята в СССР) подключить к черно-белому телевизору.

Цвет любой точки, страницы или бордюра задают кодом цвета C (от слова colour – цвет). Так, для ПЭВМ IBM PC эти коды следующие [6]:

Код	Цвет	Код	Цвет	Код	Цвет
0	Черный	5	Пурпурный	10	Светло-зеленый
1	Синий	6	Коричневый	11	Светло-бирюзовый
2	Зеленый	7	Белый	12	Светло-красный
3	Бирюзовый	8	Серый	13	Светло-пурпурный
4	Красный	9	Светло-синий	14	Желтый
				15	Ярко-синий

Для ПЭВМ, имеющих 8 цветов, используются коды:

Код	Агат	ZX-Spectrum	Код	Агат	ZX-Spectrum
0	Черный	Черный	4	Синий	Зеленый
1	Красный	Синий	5	Фиолетовый	Голубой
2	Зеленый	Красный	6	Голубой	Желтый
3	Желтый	Фиолетовый	7	Белый	Белый

Как видно из приведенных данных, цвета, кроме черного (код 0) и белого (код 7), разных ПЭВМ кодируются различными кодами C. Это следует иметь в виду при переводе программ цветной графики с одной версии Бейсика на другую.

Основные операторы задания цвета. Наиболее распространенным оператором задания цвета является оператор COLOR C. Цвет задается вводом кода C. Более гибкую конструкцию этого оператора имеет версия Бейсика Microsoft-83, применяемая в широкораспространенных ПЭВМ MSX [24]:

COLOR $C_З, C_С$ [, $C_Б$]

Здесь $C_З$ – код цвета знака; $C_С$ – код цвета страницы (знакоместа); $C_Б$ –

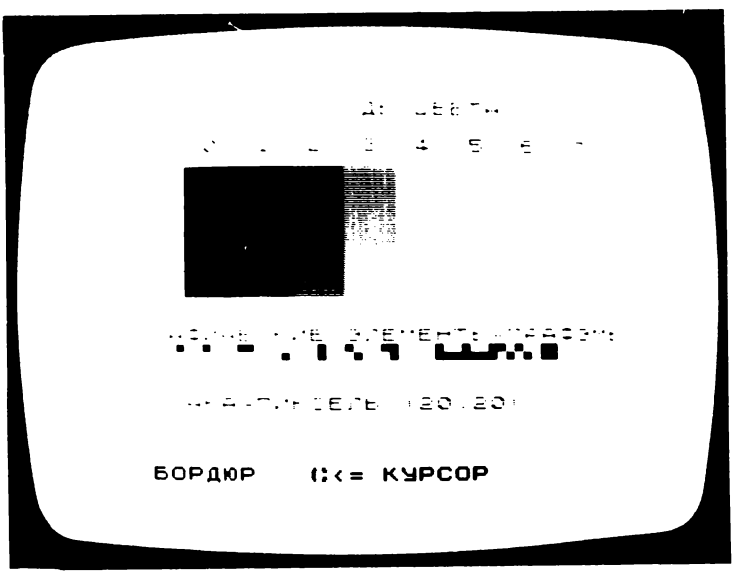


Рис. 5.2

код цвета окаймления (бордюра) экрана. У этой ПЭВМ коды 16 цветов отличаются от приведенных выше для версии Бейсика IBM PC:

Код	Цвет	Код	Цвет	Код	Цвет
1	Черный	6	Кирпичный	11	Желтый
2	Зеленый	7	Бирюзовый	12	Темно-зеленый
3	Светло-зеленый	8	Красный	13	Бледно-лиловый
4	Голубой	9	Оранжевый	14	Серый
5	Небесный	10	Темно-желтый	15	Белый

Код 0 дает "прозрачный" цвет, т.е. знак закрашивается цветом знакоместа. У ПЭВМ MSX с помощью оператора SCREEN *N* можно задавать различные режимы отображения цветной графики (моды). Они устанавливаются значением *N* (0, 1, 2 или 3).

У массовой ПЭВМ ZX-Spectrum имеются следующие операторы задания цветовых эффектов:

INK *C* (ink – чернила) – цвет знака;

PAPER *C* (paper – страница) – цвет знакоместа или всего поля экрана;

BORDER *C* (border – бордюр или окаймление) – цвет окаймления экрана;

FLASH *K* (flash – мелькать) – мерцание символов при $K=1$ (нормальный режим при $K=0$);

OVER *K* (over – покрывать) – покрытие старого изображения новым, не стирающее первое при $K=1$ (при $K=0$ нормальный режим);

INVERSE *K* (inverse – инвертирование) – инвертирование цвета знакоместа и знака при $K=1$ (при $K=0$ нормальный режим);

BRIGHT *K* – управление яркостью ($K=1$ – повышенная яркость, $K=0$ – обычная).

Здесь *C* – код цвета от 0 до 9. Код $C=8$ означает сохранение введенной цветовой функции, $C=9$ – замену цвета (при $C=0, 1, 2, 3$ цвет станет черным при $C=4, 5, 6, 7$ – белым). Нормально устанавливаются INK 0, PAPER 7 и BORDER 7 – темные знаки на белом фоне.

Указанные операторы могут быть модификаторами. Например, в предложении PRINT INK 2; FLASH 1; "COLOR": PRINT "HELLO" INK 2 (красный цвет) и FLASH 1 (мерцание) относятся только к слову "COLOR". Цвет слова HELLO будет тем, который был установлен ранее.

В ПЭВМ "Агат" оператор INVERSE используется для инвертирования цвета знакоместа. Однако возврат к нормальному отображению знакоместа осуществляется оператором NORMAL.

Системные функции задания цвета. Некоторые ПЭВМ допускают задание цвета произвольного пикселя матрицы экрана или знакоместа с помощью специальных системных функций. Более того, у простых ПЭВМ это нередко единственная возможность задания цвета. Например, у ПЭВМ Aquarius цвет любого символа и цвет знакоместа, в котором он располагается, задается оператором адресации в соответствующую ячейку ОЗУ –

POKE. Экран ПЭВМ Aquarius разбит на 24 строки и 40 столбцов, занимающих 960 ячеек ОЗУ, начиная с адреса 12328. Для вывода знака с заданным кодом в нужное знакоместо используется оператор

POKE N, Код

где N = 12328 + R*40 + C; 12328 – начальный адрес позиции в ОЗУ; R – номер строки (row); C – номер столбца (column). Например, команда

POKE 12328, 154

выводит на позицию (0,0) графему–силуэт самолета, летящего слева направо.

Матрица цветов знакомест также имеет размер 24×40 и занимает 960 ячеек ОЗУ, но других, начиная с адреса 13352. Цвета знакоместа и символа задаются оператором

$$\text{POKE } 13352 \left[, \begin{Bmatrix} \text{Код} \\ \text{цвета} \\ \text{символа} \end{Bmatrix} *16 + \begin{Bmatrix} \text{Код} \\ \text{цвета} \\ \text{знакоместа} \end{Bmatrix} \right]$$

Например, команда

POKE 13352, Ø*16 + 4

ведет к закраске знакоместа (0, 0) голубым цветом (код 4) и выведенного ранее символа черным цветом (код 0). Таким образом, эти две формы оператора POKE могут использоваться совместно. Если в последней форме опустить указание на код цвета знакоместа, он автоматически (по умолчанию) задается черным. Если не указать код цвета символа, он будет по умолчанию задан светло-зелено-голубым. У ПЭВМ Aquarius коды цвета следующие:

Код	Цвет	Код	Цвет	Код	Цвет
0	Черный	6	Светло-зелено-голубой	11	Темно-голубой
1	Красный	7	Белый	12	Светло-желтый
2	Зеленый	8	Светло-серый	13	Светло-зеленый
3	Желтый	9	Зеленовато-голубой	14	Оранжевый
4	Голубой	10	Бордовый	15	Темно-серый
5	Фиолетовый				

5.3. ПОСТРОЕНИЕ ТОЧЕК (ОПЕРАТОРЫ PSET, PRESET, PLOT, ФУНКЦИИ POINT И SCRN)

Задание координат точки. Любая точка задается в декартовой системе координат координатами x и y. Аналогично задается точка в системе координат экрана, причем x и y измеряются числом позиций экрана (одна позиция соответствует 1 пикселю). Расположение условных осей X и Y экрана у различных ПЭВМ, например "Агат" и ZX-Spectrum, может быть различным (рис. 5.3, *а* и *б*). Для вывода точки на середину экрана к x и y нуж-

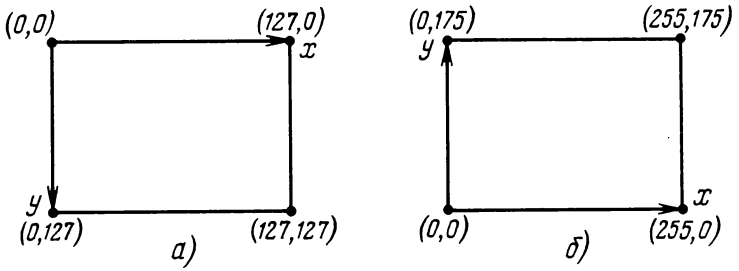


Рис. 5.3

но добавить значения x_0 и y_0, равные $n/2$ и $m/2$, где m и n – число пикселей на горизонтальной и вертикальной оси экрана.

Задание цвета точки. Цвет точки может задаваться предварительно оператором COLOR *C*. Тогда точка может иметь два состояния: высвечено (SET) и погашено (RESET). Точка с координатами x, y высвечивается при исполнении оператора

[НС] PSET(x, y) [,*C*]

Гашение точки обеспечивается оператором

[НС] PRESET(x, y)

Здесь растровые, т.е. задаваемые в виде числа пикселей растра экрана, координаты x и y могут быть константами, переменными или арифметическими выражениями. Их значения могут быть дробными, в этом случае они округляются до ближайших целых значений. Значения x и y не должны выходить за пределы: $0 \leqslant x \leqslant n-1$ и $0 \leqslant y \leqslant m-1$. У некоторых ПЭВМ допустимы отрицательные значения x и y (они воспринимаются по абсолютному значению).

Задание точки с заданными координатами обеспечивается также оператором PLOT (точка):

[НС] PLOT(x, y)

В этом случае гашение точки задается повтором этого оператора с указанием общего цвета соответствующей части экрана.

Другой способ – цвет указывается модификатором в составе оператора вывода точки. Например, у ПЭВМ ZX-Spectrum оператор PLOT может задаваться с модификаторами M1, M2 и т.д. в виде

[НС] PLOT [M1; M2; . . . ;] x, y

Модификатор INK *C* задает цвет точки, PAPER *C* – цвет знакоместа, FLASH – мигание и т.д.

Примеры.

PLOT 18Ø, 9Ø – задается (при первом пуске) черная точка с координатами $x = 180$, $y = 90$ на белом фоне экрана;

PLOT INK 2; PAPER 4; 18Ø, 9Ø – задается красная точка на фоне зеленого знакоместа с координатами (180, 90), причем цвет знакоместа задается отдельно от общего цвета страницы;

PAPER 6: PLOT INK 4; FLASH 1; 18Ø, 9Ø – задается общий желтый цвет страницы (PAPER 6) и мигающая (FLASH 1) точка с координатами (180, 90), имеющая зеленый (INK 4) цвет.

Иногда в качестве модификатора вводится указание о высвечивании (1) или гашении (0) точки:

[HC] PLOT $\begin{matrix}Ø\\ \text{или}\\ 1\end{matrix}$; x, y

Например, PLOT 1; 1ØØ, 5Ø выводит светящуюся точку с координатами $x = 100$ и $y = 50$. Цвет точки указывается предварительно. Модификатор 0 означает, что у точки устанавливается цвет экрана и она становится невидимой.

Индикация наличия точки. Часто необходимо определить, имеется ли в заданном месте экрана с координатами x, y точка, цвет которой отличен от цвета знакоместа, в котором она находится. Для этого используется функция

[HC] POINT x, y

Она принимает значение 1, если такая точка имеется, и 0, если ее нет (точнее если она имеет цвет страницы). Например, при выполнении программы

```
1Ø PAPER 7: PLOT INK 1; 18Ø, 5Ø
2Ø LET C=POINT 18Ø, 5Ø
3Ø LET D=POINT 1ØØ, 4Ø
4Ø PRINT "C="; C, "D="; D
```

значение переменной C = 1 означает, что точка с координатами (180, 50) есть и ее цвет, отличный от цвета страницы, а переменной D = 0, что точка с координатами (100, 40) имеет цвет страницы.

Функция цвета точки. Цвет любой точки экрана с координатами x, y определяется с помощью функции

[HC] SCRN (x, y)

Она принимает значение кода цвета C для указанной точки экрана с координатами x и y. Обычно применяется одна из функций: либо POINT, либо SCRN, с помощью которых осуществляется контроль положения точек, автоматическое закрашивание геометрических фигур, фиксация попадания точек в заданную область (например, в играх на попадание снаряда или пули в цель) и т.д.

5.4. ПОСТРОЕНИЕ ОТРЕЗКОВ ПРЯМЫХ И ДУГ (ОПЕРАТОРЫ PLOT–TO, DRAW, LINE)

Прямая линия

$$y = ax + b \tag{5.1}$$

может задаваться точкой (0, b) и угловым коэффициентом $a = \operatorname{tg} \alpha$ (рис.

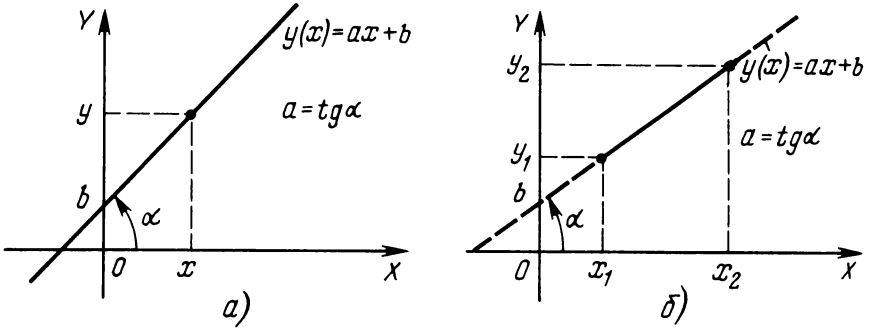


Рис. 5.4

5.4,*а*). Однако построение отрезка прямой по (5.1) неудобно, в частности, из-за необходимости вычисления наклона a. Поэтому применяют два других способа.

Построение отрезка прямой по двум заданным точкам. Пусть заданы точки начала (x_1, y_1) и конца (x_2, y_2) отрезка (рис. 5.4,*б*). Тогда уравнение прямой, проходящей через эти две точки, имеет вид

$$\frac{x-x_1}{x_2-x_1} = \frac{y-y_1}{y_2-y_1} \tag{5.2}$$

или

$$y = y_2 + (y_2 - y_1)\frac{x-x_1}{x_2-x_1}. \tag{5.3}$$

Отрезок строят, задавая x из условия $x_1 \leqslant x \leqslant x_2$. Для этого служат операторы:

[НС] PLOT x_1, y_1 TO x_2, y_2 [... TO x_n, y_n]
[НС] PLOT x_1, y_1 : LINE x_2, y_2 [... LINE x_n, y_n]

При отсутствии выражения в квадратных скобках справа операторы PLOT – TO и PLOT – LINE обеспечивают построение отрезка прямой, проходящей через точки (x_1, y_1) и (x_2, y_2). Если указанные выражения присутствуют, строится ломаная линия: точка (x_2, y_2) соединяется отрезком прямой с точкой (x_3, y_3), последняя с точкой (x_4, y_4) и т.д. Если $x_n = x_1$ и $y_n = y_1$, то строится замкнутая фигура – многоугольник.

Построение отрезка прямой по одной точке и приращениям координат. Обозначив через $\Delta x = (x_2 - x_1)$ и $\Delta y = (y_2 - y_1)$ приращение координат точки (x_2, y_2) относительно точки (x_1, y_1), получим уравнение

$$y = y_1 + \Delta y \frac{x-x_1}{\Delta x}. \tag{5.4}$$

Такой способ имеет следующие преимущества: перемещение всего отрезка на плоскости задается изменением координат не двух, а только одной точки, при вычислении y для каждого x требуется меньше арифмети-

ческих операций. Этот способ обычно реализуется оператором DRAW (тянуть):

[НС] PLOT x_1, y_1 : DRAW $\Delta x_1, \Delta y_1$ [: DRAW $\Delta x_2, \Delta y_2$: ... DRAW $\Delta x_n, \Delta y_n$,]

Если опустить предложение, стоящее в квадратных скобках справа, то обеспечивается построение прямой. В противном случае строится ломаная линия, причем в качестве начальной точки очередного отрезка используется концевая точка, вычисляемая при построении предшествующего отрезка.

Удобство этого способа заключается в том, что построенная с помощью операторов DRAW фигура может перемещаться вдоль осей x и y изменением только координат начальной точки (x_1, y_1), меньше времени затрачивается и на вычисление y по заданному x.

В обоих способах число значений x и y задается таким, чтобы построенный по точкам отрезок прямой выглядел как почти непрерывный. Однако из-за дискретности положения точек (вспомним, что каждая точка есть пиксель, расположенный в определенном месте экрана) отрезок не всегда выглядит строго как отрезок прямой. Если отрезок расположен параллельно оси X или Y, то он имеет естественный вид отрезка прямой. Однако если отрезок наклонен, то он воспринимается как ломаная линия с небольшими изломами (рис. 5.5).

Функции оператора DRAW могут быть более разнообразными. Так, у ПЭВМ MSX-основная форма записи оператора DRAW следующая:

DRAW "Строка с командами"

Строка с командами указывает на характер перемещения конца отрезка прямой, относительно начала отрезка: перемещение вверх (up), вниз (down), вправо (right) и влево (left) указывается буквами u, d, r и l, после которых ставится значение приращения в пикселях. Например, строка программы

```
10 DRAW "r50 d30 l50 u 20"
```

строит прямоугольник, имеющий размеры 50X30. Предусмотрены также перемещения конца отрезка, при которых построенный отрезок прямой имеет наклон 45, 135, 225 и 315°. Эти перемещения задаются буквами e, h, g и f. Пробелы между отдельными указаниями не обязательны. Входящие в состав оператора DRAW символы могут быть обяными символьными переменными. Оператор DRAW преобразует их в сложные геометрические фигуры.

Строка с командами может составляться из строчных переменных и констант, заключенных в кавычки, например:

```
10 A$="r50 d30"
20 DRAW A$ + "150 и 20"
```

Результат работы этого фрагмента будет аналогичен рассмотренному чуть раньше.

```
10 REM ПОСТРОЕНИЕ ПРЯМЫХ
20 PLOT 10,10: DRAW 0,100
30 PLOT 20,10: DRAW 220,0
40 FOR Y=20 TO 100 STEP 20
50 PLOT 80,20: DRAW 100,Y: NEX
T Y
```

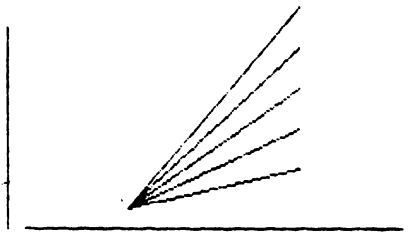


Рис. 5.5

```
10 REM ПОСТРОЕНИЕ ДУГ
20 FOR R=-4 TO 4
30 PLOT 80,50: DRAW 60,50,R
40 NEXT R
```

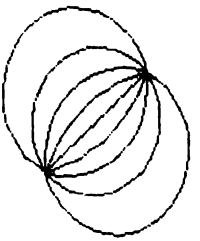


Рис. 5.6

В состав строки с оператором DRAW ПЭВМ MSX можно вводить и другие указания, например c*n* – цвет фигуры с кодом *n*, a*n* – поворот фигуры на $n \cdot 90^\circ$, s*n* – изменение масштаба в $n/4$ раз.

Таким образом, у ПЭВМ MSX возможности оператора DRAW весьма разнообразны и позволяют с его помощью строить сложные графические образы при весьма компактной форме записи программ.

Построение отрезков дуг. У ПЭВМ ZX-Spectrum оператор DRAW может использоваться для построения дуг. При этом он имеет форму

DRAW [M1; M2; ...; M*n*] $\Delta x, \Delta y$ [, r]

В его состав могут входить модификаторы M1, M2, ..., отмеченные в § 5.2. Задание параметра r обеспечивает формирование дуги, конечными точками которой являются точки (x_1, y_1) и $(x_1 + \Delta x, y_1 + \Delta y)$. Если $r = 0$, то формируется отрезок прямой линии, соединяющей точки (x_1, y_1) и $(x_1 + \Delta x, y_1 + \Delta y)$. Если $r = \pm\pi$, формируется полуокружность, если $r = \pm\pi/2$, четверть окружности и т.д. При $r > 0$ часть окружности рисуется против часовой стрелки, а при $r < 0$ – по часовой стрелке (рис. 5.6).

Следует отметить, что при $r = 0$ отрезок прямой вычерчивается гораздо быстрее (в 10–20 раз), чем дуга при $r \neq 0$. Так, у ПЭВМ ZX-Spectrum дуга строится примерно за 1 с, а отрезок прямой за 0,05 с.

Большими возможностями в построении линий обладают ПЭВМ MSX. С помощью оператора

LINE [(x_1, y_1)] – (x_2, y_2) [, C] [, B]

где C – код цвета, строится отрезок прямой, проведенной через точки (x_1, y_1) и (x_2, y_2). Если (x_1, y_1) не задано, то в качестве начальной точки отрезка берется последняя, выведенная ранее точка. С помощью модификатора STEP (шаг) в выражении

LINE [STEP(x_1, y_1)] – STEP(x_2, y_2) [, C] [, B]

координаты концевых точек указываются относительно текущего положе-

ния графического курсора. В этом случае они могут задаваться как положительными, так и отрицательными числами.

После кода цвета может стоять модификатор B (от слова box – ящик), обеспечивающий построение прямоугольников. Так, по командам PSET(1Ø0, 1Ø0) и LINE – STEP(6Ø, 6Ø) „B строится квадрат 60 X 60 с предварительно заданным цветом.

5.5. ПОСТРОЕНИЕ ЭЛЛИПСОВ И ОКРУЖНОСТЕЙ (ОПЕРАТОР CIRCLE)

Построение окружности или эллипса по каноническому уравнению. Окружность (или в общем случае эллипс) может быть построена по точкам двумя способами. В первом случае координата x каждой точки эллипса связана с координатой y известным каноническим уравнением

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, \qquad (5.5)$$

где a – радиус описанной окружности (большая полуось эллипса) и b – малая полуось эллипса (рис. 5.7). При $a = b$ эллипс вырождается в окружность.

Из (5.5) находим

$$y = \sqrt{b^2 (1 - x^2/a^2)} \qquad (5.6)$$

Задаваясь значениями x от $-a$ до $+a$, можно для каждого x вычислить два значения y. К вычисленным значениям y и заданным x надо прибавить y_0 и x_0 – координаты центра эллипса.

В эллипсе, построенном по программе, реализующей этот метод, точки расположены очень неравномерно, особенно там, где эллипс пересекает горизонтальную ось (рис. 5.8). Чтобы в этих местах предотвратить разрывы, необходимо строить кривую по очень большому числу точек, что значительно увеличивает время построения.

Построение окружности или эллипса по параметрическим уравнениям. Другой способ заключается в однозначном вычислении координат x и y точек эллипса по параметрическим уравнениям

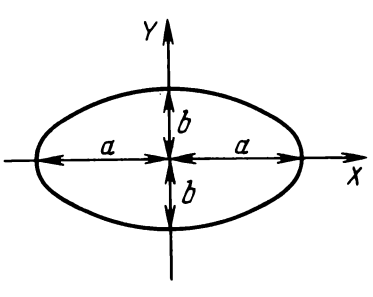


Рис. 5.7

```
  10 REM ПОСТРОЕНИЕ ЭЛЛИПСА ПО
КАНОНИЧЕСКОМУ УРАВНЕНИЮ
  20 INPUT "ВВЕДИТЕ A=";A: INPUT
"ВВЕДИТЕ B=";B
  30 FOR X=-A TO A STEP A/100
  40 LET Y=SQR (B*B*(1-X*X/A/A))
  50 PLOT X+128,Y+55: PLOT X+128
,-Y+55: NEXT X
ВВЕДИТЕ A=120
ВВЕДИТЕ B=40
```

Рис. 5.8

```
  5 REM ПОСТРОЕНИЕ ЭЛЛИПСА
 10 DIM S(360): DIM C(360)
 20 FOR I=1 TO 360: LET Q=I*PI/
180
 30 LET S(I)=SIN (Q): LET C(I)=
COS (Q): NEXT I
 40 FOR I=1 TO 360
 50 PLOT 130+80*S(I),50+40*C(I)
NEXT I
```

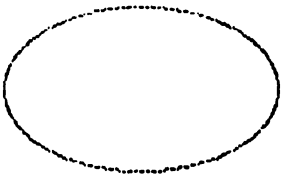


Рис. 5.9

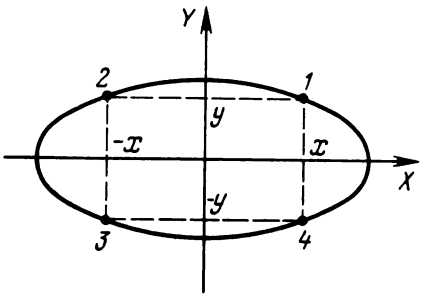


Рис. 5.10

$$x = x_0 + a\cos\varphi,\ y = y_0 + b\sin\varphi. \qquad (5.7)$$

Угол φ при этом должен меняться от 0 до 2π рад.

Главное преимущество этого способа – большая равномерность расположения точек, образующих эллипс. При $a = b = r$ эллипс превращается в окружность радиуса r :

$$x = x_0 + r\cos\varphi,\ y = y_0 + r\sin\varphi, \qquad (5.8)$$

Это справедливо при одинаковых масштабах по оси x и y экрана дисплея.

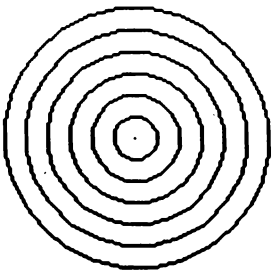
Изменение угла φ задается циклом с управляющей переменной $Q = \varphi$, меняющейся от 0 до 2π рад с шагом $2\pi/n$, где n – число точек эллипса или окружности (рис. 5.9).

Метод ускоренного построения окружности или эллипса. Эллипс с осями, параллельными осям координат, является симметричной фигурой. Это значит, что если задана точка *1* (x_1, y_1) в первом квадранте, то по ней легко сразу найти три другие точки во втором, третьем и четвертом квадрантах: *2*$(-x_1, y_1)$, *3*$(-x_1, -y_1)$ и *4*$(x_1, -y_1)$ (рис. 5.10). Применение этого принципа в программе на рис. 5.11 обеспечивает уменьшение почти в 3 раза

```
  5 REM УСКОРЕННОЕ ПОСТРОЕНИЕ Э
ЛЛИПСА
 10 DIM S(90): DIM C(90)
 20 FOR I=1 TO 90
 30 LET Q=I*PI/180: LET S(I)=SI
N (Q): LET C(I)=COS (Q): NEXT I
 40 FOR I=1 TO 90: LET X=80*S(I
): LET Y=40*C(I)
 50 PLOT 130+X,42+Y: PLOT 130-X
,42+Y: PLOT 130+X,42-Y
 60 PLOT 130-X,42-Y: NEXT I
```

Рис. 5.11

```
10 REM ПОСТРОЕНИЕ ОКРУЖНОСТЕЙ
20 FOR R=0 TO 60 STEP 10
30 CIRCLE 128,65,R: NEXT R
```



```
0 OK, 0:1
```

Рис. 5.12

числа операций вычисления тригонометрических функций (5.7). Это дает заметный выигрыш во времени при построении эллипса, поскольку тригонометрические функции $\cos\varphi$ и $\sin\varphi$ вычисляются относительно медленно.

Построение окружности с помощью оператора CIRCLE. Описанные методы построения эллипсов и окружностей основаны на использовании только оператора PLOT, присущего практически всем ПЭВМ с графикой. У ряда ПЭВМ (IBM PC, ZX-Spectrum и др.) имеется специальный оператор CIRCLE (циркуль), обеспечивающий построение окружности с центром в точке (x, y) и радиусом r. Этот оператор используется в виде

[НС] CIRCLE [M1; M2; . . . Mn;] x, y, r

где M1, M2, . . . Mn – модификаторы (см. § 5.2). Оператор CIRCLE строит окружность с помощью формул (5.8), однако все вычисления производятся микропрограммно и существенно быстрее, чем при использовании обычных программ на Бейсике. Так, на ПЭВМ ZX-Spectrum построение окружности занимает около 1 с. Применение этого оператора иллюстрирует программа, приведенная на рис. 5.12.

Наибольшими возможностями оператор CIRCLE обладает у версии Бейсика ПЭВМ IBM PC. Его общая форма записи в этой версии следующая:

[НС] CIRCLE$(x, y), a$ [, C] [,(УК1), (УК2), ␣,m/n]

Здесь x, y – координаты центра; a – наибольший радиус эллипса; С – код цвета; УК1 и УК2 – угловые координаты начальной и конечной точек дуг; ␣ – пробел; m/n – дробь, задающая эксцентриситет (по умолчанию 5/6).

Примеры.

1) 1Ø CIRCLE(16Ø, 1ØØ), 7Ø

– строится эллипс с эксцентриситетом 5/6, имеющий горизонтальную ось с растровой длиной $2 \cdot 70$, вертикальную $2 \cdot (5/6) \cdot 70$, координаты центра (160, 100) и черный цвет на белом фоне.

2) 1Ø CIRCLE(16Ø, 1ØØ), 7Ø, 2

– построение аналогично примеру 1, но эллипс красного цвета (код 2).

3) 5 CIRCLE(16Ø, 1ØØ), 7Ø, 2, (2*PI), (PI/2)

– строится дуга в виде части эллипса, начальная точка которой задается угловой координатой 2π, совпадающей с угловой координатой 0π, конечная точка угловой координатой $\pi/2$, дуга в виде четверти эллипса (см. пример 2), расположенная в первом квадранте декартовой системы координат на плоскости.

4) 5 PI=3.141592

1Ø CIRCLE(16Ø, 1ØØ), 8Ø, 2, – 2*PI, –PI/2

– указание знака минус у угловых координат означает, что концевые точки построенного эллипса (см. пример 3) окажутся соединенными отрезками прямых с точкой (x, y).

5) 1Ø CIRCLE(16Ø, 1ØØ), 8Ø, 2, 3/4

– строится эллипс с эксцентриситетом 3/4 (вертикальная ось короче горизонтальной).

6) 1Ø CIRCLE(16Ø, 1ØØ), 8Ø, 2, 2/1

– строится эллипс с эксцентриситетом 2/1 (вертикальная ось длиннее горизонтальной).

Подобными возможностями, но в несколько иной форме обладает оператор CIRCLE в версии Бейсика ПЭВМ MSX:

CIRCLE [STEP] $(x, y), r$ [, C [, a_1 [, a_2 [, a_3]]]]

Здесь x и y – координаты центра окружности (при наличии модификатора STEP приращения по оси X и оси Y); r – радиус окружности; C – код цвета; a_1 и a_2 – угловые координаты начальной и конечной точек строящейся дуги, входящей в заданную окружность; a_3 – коэффициент эксцентриситета (при построении эллипсов или эллиптических дуг). Угловые координаты начальной a_1 и конечной a_2 точек дуг задаются в радианах. Их можно перевести в градусы и наоборот с помощью соотношений: 1 рад = = 57,2958° и 1° = 0,01745 рад.

Примеры.

Применение оператора CIRCLE для версии Бейсика ПЭВМ MSX

1) CIRCLE(1ØØ, 8Ø), 5Ø

– строится окружность радиуса r = 50 с координатами центра x = 100 и y = 80; цвет окружности задается ранее введенным оператором COLOR.

2) CIRCLE(1ØØ, 8Ø); 5Ø, 8, Ø, 3.141

– строится полуокружность красного цвета (C = 8) радиуса r = 50, имеющая центр в точке (100, 80) и расположенная в первом и втором квадрантах декартовой системы координат, на что указывают значения a_1 = 0 и a_2 = 3,141 рад.

3) RC=Ø.Ø1745

CIRCLE(1ØØ, 8Ø), 5Ø, , 9Ø*RC, 18Ø*RC

– строится четверть окружности во втором квадранте с радиусом r = 50 и координатами центра x = 100 и y = 80. Угловые координаты a_1 и a_2 заданы в градусах (90 и 180°) и переведены в радианы умножением на множитель RC = 0.01745. Обратите внимание

на пропуск задания кода цвета. Это означает, что цвет дуги в четверти окружности задается последним (введенным ранее) кодом цвета.

4) CIRCLE(1Ø0, 8Ø), 5Ø, , , , 2

– строится эллипс, полученный сжатием по оси X ($a_3 = 2$) окружности с центром в точке (100, 80) и радиусом $r = 50$. В данном случае большая ось эллипса расположена вертикально и имеет длину $2r = 100$, малая ось сжата вдвое ($a_3 = 2$) и имеет длину $2r/a_3 = 50$. Пропущены код цвета (устанавливается его последним, введенным ранее значением), угловые координаты a_1 (по умолчанию $a_1 = 0$) и a_2 (по умолчанию $a_2 = 2\pi$).

5) CIRCLE(1Ø0, 8Ø), 5Ø, 8, Ø, 3.14159, .5

– строится полуэллипс, полученный из сжатой по оси Y окружности с радиусом $r = 50$ и центром в точке (100, 80) красного цвета (C = 8). Полуэллипс задан угловыми координатами $a_1 = 0$ и $a_2 = 3{,}141159$ рад ($a_2 = \pi$). На сжатие по оси X указывает значение $a_3 = 0{,}5 < 1$.

5.6. ВЫВОД И ЗАДАНИЕ ГРАФЕМ (ОПЕРАТОРЫ BIN, DRAWI, XDRAWI, SCALE, ДИРЕКТИВА GRAF)

Графемы – это графические объекты, формируемые в матрице знакоместа и хранящиеся в ПЗУ ПЭВМ (см. § 5.1). *Графемы пользователя* – графические объекты, составляемые пользователем и заносимые в ОЗУ. Графемы применяются для синтеза сложных рисунков с повышенной разрешающей способностью при малых затратах емкости ОЗУ. Вид графем зависит от назначения ПЭВМ. Универсальные ПЭВМ обычно имеют ограниченный набор встроенных графем, но предусматривают возможность задания до 10–50 графем пользователя. В ПЭВМ, ориентированных на игры (например, Aquarius), число графем достигает 200 (рис. 5.13), а в их состав входят элементы человеческих фигур, ракет, самолетов и т.д. ПЭВМ, рассчитанные на шахматные игры, имеют графемы в виде стилизованных шахматных фигур.

Вывод графем. Графемы у большинства ПЭВМ выводятся, как любые другие алфавитно-цифровые знаки. Обычно для этого служит специальная префиксная клавиша (GRAPHICS у ПЭВМ ZX-Spectrum). Они могут вводиться в текст комментария операторов REM, INPUT или PRINT. В последнем случае могут применяться модификаторы цветовых эффектов:

[HC] PRINT [M1; M2; ... Mn;] "Графемы"

Наряду с модификаторами цветовых эффектов можно использовать и модификаторы AT и TAB.

Кодирование графем пользователя. Графема пользователя задается кодированием матрицы знакоместа. На рис. 5.14 показана матрица 8×8, в которой задана греческая буква π. Светлые квадраты матрицы кодируются 0, темные 1. Таким образом, кодирование идет в двоичном коде (битах). Для кодирования каждой строки матрицы графемы применяется оператор

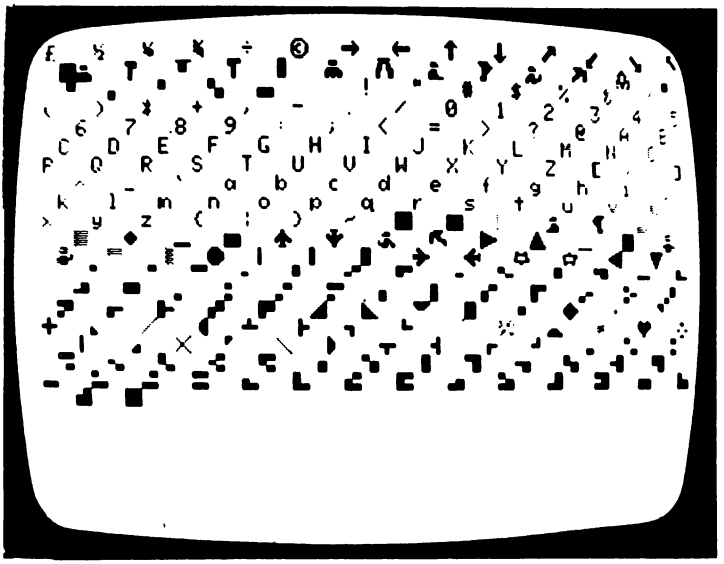


Рис. 5.13

BIN (от слова binary – двоичный). Так, графема буквы π задается восемью операторами BIN:

```
BIN  Ø Ø Ø Ø Ø Ø Ø Ø
BIN  Ø Ø Ø Ø Ø Ø Ø Ø
BIN  Ø Ø Ø Ø Ø Ø 1 Ø
BIN  Ø Ø 1 1 1 1 Ø Ø
BIN  Ø 1 Ø 1 Ø 1 Ø Ø
BIN  Ø Ø Ø 1 Ø 1 Ø Ø
BIN  Ø Ø Ø 1 Ø 1 Ø Ø
BIN  Ø Ø Ø Ø Ø Ø Ø Ø
```

Засылка кодов графем в ОЗУ. Чтобы графемы можно было использовать как любой символ, коды графемы нужно заслать в определенную область ОЗУ ПЭВМ, находящуюся под контролем соответствующей функциональной клавиши пользователя. Как правило, для этого используются специальные операторы POKE и USR, обеспечивающие доступ в ОЗУ (см. подробнее в § 3.6). Рассмотрим этот процесс на примере задания буквы π для ПЭВМ ZX-Spectrum.

Простейшая программа для построчного ввода двоичных кодов графем ПЭВМ ZX-Spectrum:

```
1Ø FOR N=Ø TO 7
2Ø INPUT B:POKE USR "P"+N, B
3Ø NEXT N
```

Здесь в строках 10 и 30 организован цикл ввода восьми элементов каждой строки. В теле цикла (строка 20) вводится бит каждого квадрата (0 или 1). Оператор POKE USR обеспечивает засылку байтов строки в область ОЗУ, находящуюся под управлением клавиши P. После ввода всех восьми строк по этой программе нажатие клавиши P в режиме GRAPH будет вызывать на индикацию букву π.

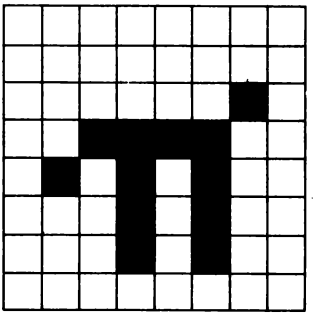


Рис. 5.14

Вместо оператора BIN с аргументом можно вводить непосредственно его десятичное значение. Тогда графемы задаются восемью десятичными кодами (байтами), кодирующими ее строки. Например, буква π задается кодами: 0, 0, 2, 60, 84, 20, 20, 0. Такая форма ввода менее наглядна (в поле аргументов оператора BIN единицы дают явное изображение графемы), но более компактна.

С помощью описанной программы, заменяя "P" буквами от "A" до "U", можно ввести в ПЭВМ ZX-Spectrum графемы 21 буквы русского алфавита и затем записать их на магнитофон командой

SAVE "RALE" CODE 65368, 168

Коды занимают в ОЗУ пространство $21 \cdot 8 = 168$ байт, начиная с адреса 65368. В дальнейшем их можно считывать с магнитофона командой LOAD "RALF" CODE и применять в программах, например, для задания комментариев на русском языке. С помощью специальных программ можно создать до 500 графем.

Еще большие возможности задания графем имеет ПЭВМ MSX. В ней можно задавать до 64 графем, определяя их набор оператором

SCREEN *M, N*

где M – номер режима (моды) вывода графем; N – число графем. Графемы при $M = 0$ задаются матрицей 8×8 бит (или 8 байт), при $M = 1$ заданные так же выводятся увеличенными (по размеру) вдвое. При $M = 2$ графемы задаются матрицей 16×16 бит (32 байт), а при $M = 3$ заданные так же выводятся увеличенными вдвое.

Образ графемы задается командой

SPRITES (I) = ⟨Строчная переменная⟩

где I – присвоенный графеме номер ($I \leqslant N \leqslant 64$).

Преобразование двоичного кода графем в строчную переменную выполняется оператором

CHR$ (& B⟨Двоичный код строки графемы⟩)

Например, буква π в виде графемы с номером $I = 5$ задается следующим образом:

```
200 B0$=CHR$ (& B 00000000)
210 B1$=CHR$ (& B 00000010)
```

```
220  B2$=CHR$ (& B 00111100)
230  B3$=CHR$ (& B 00010100)
240  S1$=B0$+B0$+B1$+B2$+B3$+B3$+B3$
250  SPRITES(5)=S1$
```

Вывод сформированной графемы в нужное место экрана задается оператором

PUT SPRITE *I*, [STEP(*x*, *y*)] [, C]

где *I* – номер графемы; C – код ее цвета. Если модификатор STEP (*x*, *y*) пропустить, т. е. записать оператор вывода графемы в виде

PUT SPRITE *I*, , , C

то графема будет выведена на место курсора. Если опустить только слово STEP, то оператор

PUT SPRITE *I*, (*x*, *y*), C

обеспечит вывод графемы на место с координатами *x* и *y*, заданными в пикселях. Наконец, в общем виде этот оператор выводит графему с номером *I* и цветом C на место, расположенное на расстоянии *x* по оси *X* и *y* по оси *Y* относительно положения курсора.

Таким образом, имеются средства вывода графем в наиболее удобной для пользователя форме, как с непосредственной адресацией положения графемы на экране, так и с адресацией относительно графического курсора.

У некоторых ПЭВМ (Агат, Apple-II) выполняется кодирование графем (образцов) в точке с координатами *x*, *y* в текущем цвете с помощью оператора

DRAWI AT *x*, *y*

и в дополнительном цвете

XDRAWI AT *x*, *y*

Правила кодирования более сложные и их описание можно найти в технических руководствах по эксплуатации этих ПЭВМ. Полученные образцы можно менять в масштабе с помощью оператора SCALE=*X* и поворачивать угол, заданный оператором ROT=*X*.

5.7. ДОПОЛНИТЕЛЬНЫЕ ГРАФИЧЕСКИЕ ОПЕРАТОРЫ И ФУНКЦИИ РАСШИРЕННЫХ ВЕРСИЙ ЯЗЫКА БЕЙСИК (ALTER, BOX, DRAW TO, CIRCLE, PAINT, FILL, WINDOW, OPEN, CLOSE, BLOCK, POINT, CURSOR, ARC, XOS, YOS, XRG, YRG, SCRN$)

Многие версии Бейсик допускают расширения с помощью дополнительных программ, обычно записываемых в машинных кодах. Такие расширения порождают новые операторы, существенно повышающие графические возможности Бейсика [54–56].

Оператор перемены атрибутов графических изображений

ALTER ⟨Атрибуты⟩ TO ⟨Атрибуты⟩

```
10 PLOT 0,10: DRAW 255,0
20 PLOT 0,130: LET y=70
30 FOR X=2 TO 250 STEP 2
40 LET Y1=10+60*COS (X/10)
50 LET DY=Y1-Y: LET Y=Y1
60 DRAW 2,DY
70 NEXT X
```

sin(x)

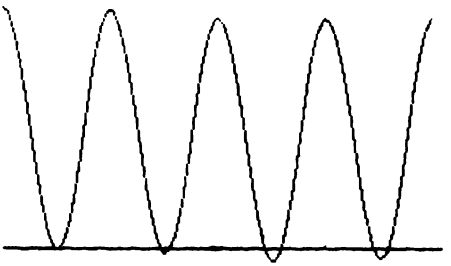


Рис. 5.15

позволяет сменить одни атрибуты на другие, допускает различные применения:

ALTER TO PAPER 1 – для всех знакомест устанавливается цвет страницы с кодом C = 1;

ALTER TO PAPER 6, INK 1, FLASH 1 – для всех знакомест устанавливается цвет страницы кода 6, цвет изображения кода 1 и режим мерцания кода 1;

ALTER INK 0 TO INK 7 – для всех позиций, имеющих цвет изображения кода 0, устанавливается цвет кода 7.

Упомянутый ранее оператор DRAW имеет следующий недостаток: при построении с его помощью ломаных линий с большим числом отрезков прямых или дуг постепенно накапливается погрешность вычисления координат x и y по их приращениям Δx и Δy, так как координаты округляются до целых чисел. В результате линия может несколько отклоняться от расчетных координат, заданных прямым указанием x и y. Возникновение отмеченного недостатка поясняет программа, приведенная на рис. 5.15. Эта программа строит отрезок прямой с координатами точек (0,10) и (255,10) и косинусоиду $y = 10 + 60\cos(x/10)$. Последняя строится с помощью малых отрезков, формируемых оператором DRAW. Вершины отрицательных полуволн синусоиды должны касаться отрезка прямой, но не пересекать его. В действительности этого нет и наблюдается постепенное сползание косинусоиды вниз, заметно искажающее график функции (рисунок под программой).

Этот недостаток отсутствует у расширения оператора DRAW:

DRAW TO x, y [, r]

Данный оператор проводит отрезок прямой (если $r = 0$) или дуги (если $r \neq 0$), проходящий через последнюю установленную точку с координатами (x_0, y_0) и новую точку с координатами (x, y). При первом применении оператора DRAW TO считается, что предшествующая точка имеет координаты (0,0).

Программа, приведенная на рис. 5.16, функционально повторяет описанную выше, но использует оператор DRAW TO. Применение оператора DRAW TO упрощает программу: нет необходимости в вычислении приращений и хранении для этого предшествующих значений функций. В этом случае сползание графика функции отсутствует и косинусоида лишь касается вершинами отрицательных полуволн отрезка прямой.

```
10 PLOT 0,10
   DRAW 255,0
20 PLOT 0,130
30 FOR x=2 TO 250 STEP 2
40   LET Y=70+60*COS (X/10)
50   DRAW TO X,Y
   NEXT X
```

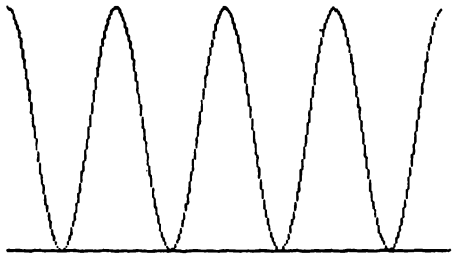


Рис. 5.16

Встречаются также различные варианты оператора LINE, например: LINE x, y – строит вектор, проходящий через точку (0,0) и точку (x, y), LINE $(x_1, y_1) - (x_2, y_2)$ – строит отрезок прямой, проходящей через точки (x_1, y_1) и (x_2, y_2).

Оператор MOVE x, y используется для установки точки с координатами (x, y), которая служит опорной точкой для построения других фигур. Так, оператор

BOX (w, h)

строит прямоугольник с центром в точке (x, y) шириной w и высотой h. Оператор

CIRCLE R, N

строит эллипс с длиной вертикальной полуоси $2R$ и коэффициентом эллиптичности N, N может отличаться от принятого в математике определения. Например, при $N=10$ формируется окружность, при $N<10$ получаем эллипс, сплюснутый по горизонтали, а при $N>10$ – растянутый по горизонтали.

Оператор

PAINT [Атрибуты, x, y]

служит для закраски замкнутых фигур. Если атрибуты закраски не указаны, используются ранее действующие атрибуты. Координаты точки (x, y), с которой начинается закраска, устанавливаются любыми в пределах закрашиваемого участка с помощью оператора MOVE x, y или с помощью аргументов в самом операторе PAINT. Например,

MOVE 12Ø, 8Ø: INK Ø: BOX 4Ø, 2Ø: PAINT

– строится прямоугольник, который закрашивается черным цветом (INK Ø).

Более гибким является оператор закраски замкнутых фигур

FILL [Атрибуты ;] x, y

где координаты начальной точки закраски (x, y) используются как входные параметры. Действие оператора FILL [54] иллюстрирует программа, приведенная на рис. 5.17. По этой программе (строки 10–40) строится окружность, которую частично пересекает отрезок горизонтальной прямой. При пуске программы получим первую фигуру, которая в результа-

```
10 CIRCLE 120,80,60
20 PLOT 120,100
30 DRAW 100,0
40 PAUSE 0
50 FILL INK 0;150,80
```

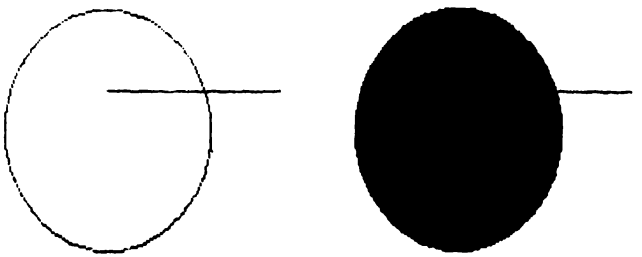


Рис. 5.17

те выполнения оператора закраски FILL (строка 50) целиком закрашивается.

Обычно алгоритм закраски сводится к перемещению отдельного элемента закраски по строкам или столбцам внутри замкнутой фигуры с проверкой наличия незакрашенных областей и изменением направления закраски при их наличии. Такой алгоритм обеспечивает закраску фигур, подобных приведенной. Однако в некоторых версиях Бейсика операторы закраски PAINT и FILL реализуют упрощенный алгоритм закраски с перемещением элемента закраски только по строкам или только по столбцам. В этом случае могут образоваться незакрашенные внутренние области и придется повторить ввод операторов закраски, указав новую начальную точку в такой области. Если исходная точка закраски выбрана за пределами замкнутой кривой, то внешней границей закраски считается окаймление экрана, и будет закрашиваться вся область экрана между незакрашенной замкнутой кривой и окаймлением экрана.

В технике отображения информации на экране дисплея ПЭВМ широкое распространение получили так называемые окна – отдельные прямоугольные участки экрана, в которых создаются независимые друг от друга изображения. Например, в одном окне можно получить изображение самолета, а в другом – карту местности, над которым он пролетает. Представление различной информации в окнах значительно увеличивает ее объем, воспринимаемый пользователем. Удобно использовать окна и для наглядного графического представления меню операций, выполняемых ПЭВМ.

Для создания окон чаще всего применяется оператор WINDOW (окно) вида

WINDOW N [, x, y, h, w]

где N – номер окна; x и y – координаты опорного угла; h, w – высота и ширина окна. Если x и y не заданы, полагается $x = 0$ и $y = 0$. Если h и w не заданы, окном задается весь экран. Обычно опорным является левый нижний угол прямоугольника, задающего размеры окна.

```
  5 CIRCLE 120,70,60
 10 WINDOW 1,10,90,80,80
 20 WINDOW 2,50,130,80,80
 30 WINDOW 3,90,170,80,80
 40 PRINT WINDOW 1;"01234567890
12345678901234567890123456789012
34567890123456789012345678901234
56789012345"
 50 PRINT WINDOW 2;"XXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXX"
 60 PRINT WINDOW 3;"00000000000
00000000000000000000000000000000
00000000000000000000000000000000
0000000000000000"
 75 CIRCLE 129,135,30
```

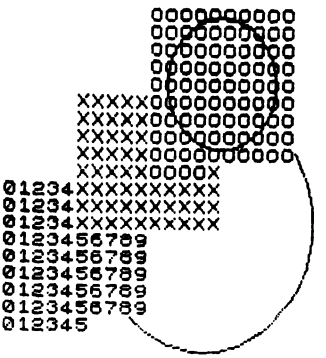


Рис. 5.18

В окне могут создаваться самые различные изображения: тексты, графики, графемы и т. д. Чаще всего координаты окна относятся лишь к тексту, и графики можно создавать независимо. Если график создается раньше окна, то окно окажется наложенным на него и приведет к исчезновению части графика. И напротив, график, построенный позже окна, будет наложен на формируемое в окне изображение. Все эти случаи иллюстрирует программа на рис. 5.18. В ней задаются три окна с различными текстами в них (цифры, буквы X и 0). Окна частично пересекаются и накладываются друг на друга. Кроме того, до построения окон строится окружность большого радиуса, а после этого – окружность малого радиуса. Получаемая при этом (с помощью принтера с применением команды COPY) картина показана рядом с программой.

В оператор WINDOY можно вводить модификаторы для задания цветовых эффектов. Например,

WINDOW 1: PAPER *M*: INK *N*

– задает цвет страницы с кодом *M* и знаков с кодом *N* для окна 1. Стирание окна выполняется командой CLS *N*, где *N* – номер окна.

У версии Бейсика QL SUPERBASIC ПЭВМ QL [52] окна задаются командой

WINDOW# *N*, *w*, *h*, *x*, *y*

Имеется также ряд дополнительных команд:

OPEN# *N*, SCR_ *w***hA x***y* – открывает окно с номером *N*,

CLOSE# *N* – закрывает окно с номером *N*,

BLOCK *w*, *h*, *x*, *y*, *c* – задает блок с нужным цветом (код C) страницы. Например, команда

OPEN# 1, SCR_ 4Ø*4Ø *A* 12Ø*1ØØ

открывает окно 40X40 с номером 1 и координатами нижнего левого угла (120, 100);

PRINT# 1, "HELLO"

выводит слово "HELLO" в окно с номером 1.

В этой версии Бейсика можно отдельно задавать координаты в графических операторах для любого окна. Для этого в операторы вводится модификатор #*N*, где *N* – номер окна. Например (в них указаны дополнительные операторы POINT, CURSOR и ARC):

POINT# 1, 4,4 TO 1Ø, 1Ø – протяжка линии через точки (4,4) и (10,10) в окне 1,

CURSOR# 2, 1Ø,1Ø,5Ø,5Ø – установка курсоров (графического и обычного) в точки (10,10), и (50,50) окна 2,

CIRCLE# 3, X, Y, R – построение окружности с координатами (X, Y) и радиусом R в окне 3,

ARC# 4, X1, Y1 TO X2, Y2, ANGLE – построение дуги в окне 4, проходящей через точки (x_1, y_1) и (x_2, y_2) с углом поворота ANGLE.

При отсутствии модификатора #*N* заданные координаты относятся к экрану в целом.

В расширенной версии Бейсика Mega-BASIC [55] для открытия окна с номером *N* используется команда

CURRENT_*N*

где *N* – номер окна. Само окно задается последующей командой

WINDOW_*x, y, h, w*

где x и y – координаты левого верхнего угла; h, w – высота и ширина окна. Команда

CLW_*N*

обеспечивает быструю очистку окна *N* от изображения. Данная версия Бейсика обладает возможностью задания процедур: @ Имя – объявление процедуры, ENDPROC_ Имя – закрытие процедуры, организации циклов REPEAT – UNTIL_ Условие и рядом других отмеченных далее особенностей, полезных для создания сложных графических объектов (в том числе динамических).

В расширенные версии Бейсика входят также системные функции. Они позволяют судить о работе ПЭВМ или менять ее отдельные характеристики.

Системные переменные XOS и YOS [54] задают начало отсчета в декартовой системе координат экрана (их значения выражаются в пикселях). Переменные XRG и YRG задают изменение масштаба для операторов PLOT и DRAW.

Допустим, например, что значение координаты x меняется от 0 до 255. Этому соответствует XOS = Ø. Если задать

LET XOS = 1ØØ

то координате x можно придавать значения от –100 до +155. Таким образом, начало координат смещается (в данном примере вдоль оси *X* на 100 пикселей).

```
  10 GO SUB 100: REM NORMAL
  20 LET XRG=128: GO SUB 100
  30 LET YRG=88: GO SUB 100
  40 LET XRG=256: GO SUB 100: ST
OP
 100 REM TRIANGLE
 110 PLOT 0,0: DRAW 100,0
 120 DRAW -50,80: DRAW -50,-80
 130 PAUSE 0: RETURN
```

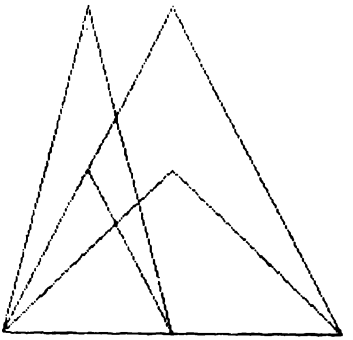


Рис. 5.19

Влияние переменных XRG и YRG на изменение масштаба по осям X и Y при построении 4 треугольников иллюстрирует программа на рис. 5.19.

Во всех случаях треугольник строится по подпрограмме (строки 100–130). В строке 10 задано построение исходного треугольника. Нажав любую клавишу, переходим к построению треугольника с уменьшенным вдвое масштабом по оси X (строка 20). Нажав снова любую клавишу, получим треугольник с уменьшенным вдвое масштабом по осям X и Y (строка 30). Наконец, еще раз нажав любую клавишу, получим треугольник с восстановленным масштабом по оси X, но уменьшенным вдвое по оси Y. Итоговая картина показана под программой. Отметим, что уменьшение масштаба по осям X и Y означает увеличение размеров изображения.

Функция SCREEN$ (m, n), где m и n – номер строки и столбца, вырабатывает символ знака, расположенного в указанной позиции, если этот знак не есть графема (в последнем случае эта функция ничего не вырабатывает). Дополнительная функция SCRN$ (m, n) аналогична функции SCREEN$ (m, n), но позволяет выработать и символ графемы.

Функция FILLED вырабатывает число точек, которое использовано оператором закраски FILL (в него не входят точки контура замкнутой фигуры). Код атрибутов знакоместа выдает функция ATTR.

5.8. УПРАВЛЕНИЕ ШРИФТОМ (ОПЕРАТОРЫ CSIZE, SHIFT$, MODE, CHR$, FONT

К специальным призмам машинной графики относится создание различных шрифтов. Чтобы в строке поместилось больше чисел или слов лучше выдавать результаты мелким шрифтом. Мелкий шрифт удобен и при печати индексов у математических символов. Для заголовков программ нагляднее использовать крупный шрифт, особенно для учебных и игровых программ. В связи с этим расширенные версии Бейсика содержат специальные графические средства для управления шрифтом – его размерами, а иногда и типом.

В расширении Бейсика Beta-BASIC 3.0 и в QL SUPERBASIC [52, 54] для управления шрифтом используются директива CSIZE и символьная функция SHIFT$. Директива CSIZE записывается в виде

CSIZE m [,n]

```
10 CSIZE 4,8: PRINT " CSIZE 4,
8"
20 CSIZE 8,8: PRINT " CSIZE 8,
8"
30 CSIZE 8,16: PRINT " CSIZE 8
,16"
40 CSIZE 16,16: PRINT " CSIZE
16,16"
50 PRINT CSIZE 16,32;" CSIZE 1
6,32"
60 PRINT CSIZE 24,32;" CSIZE 2
4,32 Hello!"
70 CSIZE 8,8
```

```
CSIZE 4,8
CSIZE 8,8
CSIZE 8,16
 CSIZE 16,16
CSIZE 16,32
 CSIZE 24,
32 Hello!
```

Рис. 5.20

где m – ширина знака, а n – его высота. Значение CSIZE 8 эквивалентно CSIZE 8,8 и соответствует нормальному шрифту.

Следующая программа, приведенная на рис. 5.20, дает образцы шрифтов при использовании директивы CSIZE m, n для разных m и n. Как видно, директива CSIZE при увеличении размеров знаков просто копирует исходный знак. Однако при уменьшении шрифта автоматически меняется написание отдельных знаков, например нуль выводится уже неперечеркнутым. Директива CSIZE может быть модификатором в команде PRINT.

Символьная функция

SHIFT$ (N, Строка)

создает строку, полученную преобразованием заданной строки (или символьной переменной). Вид преобразования задается кодом N (до N=11). Например, при N=1 все малые буквы строки становятся большими, при N=2 все большие буквы строки – малыми, при N=3 малые буквы становятся большими, а большие – малыми и т.д.

Действие функции CHIFT$ для N=1, 2 и 3 поясняет программа на рис. 5.21. Результат ее выполнения представлен под программой – слева исходная строка (слово Basic), справа – результат преобразования.

Обширными возможностями задания шрифтов обладает расширение Mega-BASIC [55]:

MODE_M – задание общего режима (моды) отображения;

CHR$_$M$ – модификатор в операторе PRINT, задающий вид шрифта в пределах действия этого оператора;

```
10 LET A$="Basic"
20 LET B$=SHIFT$(1,A$)
30 PRINT A$,B$
40 LET C$=SHIFT$(2,A$)
50 PRINT A$,C$
60 LET D$=SHIFT$(3,A$)
70 PRINT A$,D$

Basic                BASIC
Basic                basic
Basic                bASIC
```

Рис. 5.21

```
  5 FONT_0:MODE_1: CLS : PAPER
7: INK 0
  10 PRINT CHR$ 0;"CHR$0"
  15 PRINT CHR$ 1;"CHR$1"
  20 PRINT CHR$ 2;"CHR$2"
  25 PRINT CHR$ 3;"CHR$3"
  30 PRINT CHR$ 4;"CHR$4"
  40 PRINT CHR$ 5;"CHR$5"
  50 FONT_0: PRINT "Mega BASIC"
  55 FONT_1: PRINT "Mega BASIC"
  60 FONT_2: PRINT "Mega BASIC"
  70 FONT_1: PRINT CHR$ 2;"FONT_
1:PRINT CHR$2"
```

```
CHR$1
CHR$2
CHR$3
CHR$4
CHR$5
Mega BASIC
Mega BASIC
Mega BASIC
FONT_1:PRINT CHR$2

0 OK, 70:2
copy
```

Рис. 5.22

FONT_*M* – установка типа шрифта (действует и на вывод оператором PRINT и на листинг программы).

Действие всех этих операторов при различных *M* иллюстрирует программа на рис. 5.22, рядом показаны образцы шрифтов. Отметим, что, задавая различные *M* в командах MODE, CHR$ и FONT, можно получить множество дополнительных шрифтов разных размеров и вида.

У ряда ПЭВМ (включая вычислительные комплексы серии ДВК) управление шрифтом осуществляется с помощью специальной программы, вызываемой монитором. Имеются также специальные программы (например, BIGWRITE – писание большими буквами), создающие различные шрифты. В их основе лежит техника создания шрифтовых графем по исходному образцу. Так, для увеличения шрифта каждый пиксель исходного образца представляется матрицей из $m \times n$ пикселей в блоке (m и n – число строк и столбцов блока). Имеются такие программы на Бейсике, но они работают очень медленно. Поэтому чаще программы задания шрифта пишутся на машинных кодах (именно так реализовано изменение шрифта и в специальных командах расширений Бейсика, описанных выше).

5.9. ДИНАМИЧЕСКАЯ ГРАФИКА (ОПЕРАТОРЫ ROLL, SCROLL, PAN, FADE, GET, PUT, PLOT, *GRAPHIC, *CHAR, *SPRITE И ДР.)

Элементами динамической графики являются подвижные изображения, создаваемые на экране дисплея. Это могут быть тексты и надписи, "плывущие" по экрану и постепенно исчезающие или различные графические объекты (фигурки людей и животных, технические аппараты и др.). Динамическая графика широко используется в учебных и игровых программах. В сложных версиях Бейсика и его расширениях имеется ряд команд, облегчающих создание динамической графики. Типовые из них описаны ниже.

Команда вращения изображения ROLL задается в виде [54]

ROLL *K* [,*N*; *x*, *y*; *w*, *l*]

Здесь *K* – код направления; *N* – число точек единичного перемещения; *x* и *y* – координаты перемещаемой части изображения; *w* и *l* – ширина и длина (в знакоместах) этой части. Заданный параметрами *x*, *y*, *w* и *l* блок

изображения перемещается в любом из 4 направлений, что задается кодом K: 5 – влево, 6 – вниз, 7 – вверх и 8 – вправо. Перемещение, если его повторять, имеет характер вращения, т.е. исчезающий с одной стороны фрагмент изображения появляется с другой стороны. Команда ROLL K обеспечивает единичное перемещение всего изображения на 1 пиксель в направлении, указанном кодом K.

Чтобы создать эффект плавного перемещения изображения нужно повторить команду ROLL, например, в цикле. Например, программа

```
   5 CIRCLE 120,80,10: CIRCLE 12
0,80,70
  10 FOR D=5 TO 8
  20 FOR P=1 TO 100
  30 ROLL D
  40 NEXT P: NEXT D
```

строит две концентрические окружности, которые затем плавно перемещаются влево, вниз, вверх и вправо, в конце занимают исходную позицию.

Координаты блока x и y задаются в пикселях, w и l – в знакоместах (но отличных от нормальных); w могут принимать значения от 1 до 32, а l от 1 до 176. Следующая программа иллюстрирует сложное перемещение изображения сразу в 4 блоках:

```
 200 LIST 200 TO 260: LIST 200 T
O 260: LIST 200 TO 260
 210 LET P=4
 220 ROLL 5,P;0,175;32,88
 230 ROLL 6,P;0,175;16,176
 240 ROLL 8,P;0,87;32,88
 250 ROLL 7,P;128,175;16,176
 260 GO TO 220
```

Изображение в блоках перемещается и перемешивается, создавая эффект вращения против часовой стрелки. Начальный кадр и стоп-кадр изображения через несколько секунд показаны на рис. 5.23,*а* и *б*.

Другая команда

SCROLL [K,N: x,y; w,l]

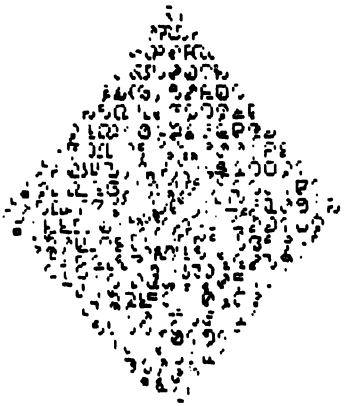
подобна команде ROLL. Отличие ее в том, что исчезающий участок изображения уже не появляется с противоположной стороны, а исчезает с поля зрения (экрана). Эта команда в форме SCROLL (без аргументов) вызывает скачок всего изображения вверх на 1 строку. Действие команды SCROLL можно проверить с помощью программы, подобной приведенной выше, но команда ROLL заменена SCROLL:

```
 200 LIST 200 TO 260: LIST 200 T
O 260: LIST 200 TO 260
 210 LET P=4
 220 SCROLL 5,P;0,175;32,88
 230 SCROLL 6,P;0,175;16,176
 240 SCROLL 8,P;0,87;32,88
 250 SCROLL 7,P;128,175;16,176
 260 GO TO 220
```

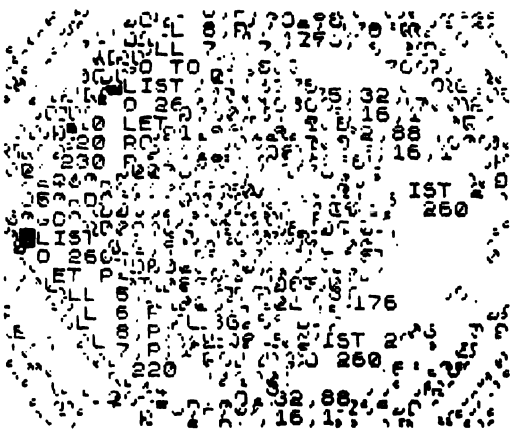
Исходное изображение на экране соответствует рис. 5.23,*а*, через несколько секунд появится изображение, подобное показанному на рис. 5.23,*в*.

```
200 LIST 200 TO 260: LIST 200 T
    O 260: LIST 200 TO 260
210 LET P=4
220 ROLL 5,P,0,175,32,88
230 ROLL 6,P,0,175,16,176
240 ROLL 8,P,0,87,32,88
250 ROLL 7,P,128,175,16,176
260 GO TO 220
200 LIST 200 TO 260: LIST 200 T
    O 260: LIST 200 TO 260
210 LET P=4
220 ROLL 5,P,0,175,32,88
230 ROLL 6,P,0,175,16,176
240 ROLL 8,P,0,87,32,88
250 ROLL 7,P,128,175,16,176
260 GO TO 220
200 LIST 200 TO 260: LIST 200 T
    O 260: LIST 200 TO 260
210 LET P=4
220 ROLL 5,P,0,175,32,88
230 ROLL 6,P,0,175,16,176
240 ROLL 8,P,0,87,32,88
```

а)



в)



б)

Рис. 5.23

Несколько иначе перемещения изображения организованы в расширении Mega-BASIC [55]. Здесь команда PAN_N, $\pm X$ перемещает изображение вдоль оси X на величину $\pm X$, а команда SCROOL_N, $\pm Y$ – то же, но по оси Y. Перемещения относятся к окну с номером N. Команда динамического стирания изображения FADE_N создает оригинальные световые эффекты стирания изображения (мерцание, быстрая смена цветов и постепенное угасание изображения), применяется, например, для имитации катастроф (нечто подобное, видимо, видит попавший в катастрофу в последние мгновения перед потерей сознания).

Ряд команд используется для создания динамических объектов, передвигающихся по экрану с достаточно большой скоростью. Даже у простых версий Бейсика предусмотрено создание графем, помещаемых на нужное знакоместо экрана (см. § 5.6). Выводя последовательно соответствующие графемы, можно синтезировать движущиеся объекты (фигуры людей, самолетов и т. д.), однако подобное движение (по знакоместам) выглядит не непрерывным. Объекты прыгают из одного знакоместа в другое, и эти скачки хорошо видны на глаз. В этой связи в расширенных версиях Бейсика используются специальные приемы для создания образов объектов – спрайтов, их деформации (перемещения, вращения и т. д.) и возможности быст-

рого вывода в любую точку экрана с предельным разрешением в 1 пиксель (а не в одно знакоместо). В простейшей форме техника задания спрайтов для ПЭВМ MSX была описана в § 5.6. Ниже описываются дополнительные возможности задания и управления спрайтами в других версиях языка Бейсик.

В версии Бейсика ПЭВМ IBM PC для создания образов используется оператор GET в виде [53]

GET (x_1, y_1) (x_2, y_2), Имя

Здесь x_1, y_1 – координаты верхнего левого; x_2, y_2 – координаты нижнего правого угла блока изображения; Имя – имя массива. Оператор GET создает числовой массив с указанным именем, хранящий всю информацию об изображении в прямоугольном блоке, заданном координатами (x_1, y_1) и (x_2, y_2) его углов. Само изображение должно формироваться любым из описанных ранее способов: вывод текста, графем, применение операторов PSET, LINE, CIRCLE, DRAW, PAINT и др. Таким образом, оператор GET как бы снимает и запоминает копию произвольного изображения со всеми его атрибутами.

Оператор

PUT (x, y), Имя

выводит блок изображения с указанным именем хранящего его массива в заданное место экрана с координатами (x, y) верхнего левого угла. Техника применения операторов GET и PUT подробно описана в [53], пример дан в приложении 2.

Почти аналогично создаются образы в расширении Beta BASIC 3.0 [54]. Для этого используется оператор:

GET Имя, x, y [, w, l] [; Тип]

Здесь Имя – имя строчной переменной, хранящей данные об образе; x и y – координаты левого угла блока образа (в пикселях); w (widh) – ширина и l (lenght) – длина блока (в знакоместах); Тип (число 0 или 1) – указание на один из двух возможных типов задания расцветки образов. Хранение данных в виде значения строчной переменной имеет важное достоинство – нет необходимости в резервировании под данные памяти, так как оно происходит автоматически.

Вызов образа-спрайта выполняется командой

PLOT x, y, Имя

где x и y – координаты верхнего левого угла выводимого образа (в пикселях); Имя – имя символьной переменной, хранящей образ.

Действие этих операторов иллюстрирует программа, приведенная на рис. 5.24. Строится закрашенный круг с выведенной в его середину надписью HELLO (строка 10). Оператор GET копирует этот рисунок, подрезая круг снизу (строка 20). С помощью оператора PLOT подрезанный круг перемещается по оси X заданием переменной координаты x (строки 30–50). Ниже представлен стоп-кадр изображения: в левом верхнем углу исход-

```
10 CIRCLE 40,140,30
   FILL 40,140
   PLOT 20,144,"HELLO"
20 GET X$,10,175,8,8
30 FOR X=0 TO 255
40   PLOT X,80,X$
50 NEXT X
```

Рис. 5.24

ное изображение (спрайт), ниже (в центре экрана) – его перемещающаяся подрезанная снизу копия. (Принтер, с помощью которого получены рисунки, несколько вытягивает изображение по вертикали, так что окружность выглядит как эллипс.)

Скорость движения можно увеличить, задав изменение с большим шагом (принят шаг 1). Однако с ростом шага сильнее проявляется дискретность перемещения (в виде скачков). Напротив, замедлить движение можно, введя после оператора PLOT команду паузы PAUSE *N*, где с ростом *N* движение будет замедляться.

Скорость перемещения спрайтов в описанных выше версиях сравнительно невелика. Например, в последнем примере объект перемещается вдоль экрана примерно за 10 с (шаг изменения *x* равен 1). Разработаны, однако, специальные версии Бейсика с ускоренной динамической графикой. Примером может служить версия Sprite BASIC (фирма Mapasoft, Польша, 1986 г.) [56]. Ее особенности описаны ниже.

Команды версии Sprite-BASIC вводятся со знаком * перед ними. В составе версии имеется ряд ранее описанных команд, представляемых в несколько ином виде:

*DOKE Адрес, Код – двойное POKE,
*REPEAT: . . . : *UNTIL – цикл типа REPEAT – UNTIL,
*SET INK C1, C2 – замена кода цвета знака C1 на код C2,
*SET PAPER C1, C2 – замена кода цвета страницы C1 на C2,
*SCREEN PAPER C1, INK C2 – установка цвета страницы (код C1) и знаков (код C2),
*CRAPHIC *N* – задание процедуры ввода графического элемента,
*PUT *N, x, y* – вывод образа с номером *N* на место с координатами опорного угла (*x, y*),
*ERASE ∅ – стирание всего экрана,
*ERASE SPRITE *N* – стирание спрайта с номером *N*,
*SPRITE *N, x, y* – вывод спрайта с номером *N* на место с координатами опорного угла *x, y*,
*SPRITE *N, x, y, x – w, y – w, K* – вывод спрайта с дополнительными атрибутами перемещения,
*CHAR *N*, S1, S2, S3, S4 – задание процедуры создания динамического спрайта с номером *N*; имеющего изображение в виде спрайтов с номерами S1, S2, S3 и S4,

*DIN . . . *E1 – объявление начала и конца процедуры задания подвижного спрайта (см. команду *CHAR).

С помощью оператора

*GRAPHIC *N*

где *N* – целое число от 1 до 32, можно задать до 32 спрайтов, каждый из которых строится в расширенной матрице (16×16 пикселей). Для этого сразу после оператора *GRAPHIC должен следовать оператор *DATA со списком из 16 чисел, каждое из которых кодирует одну строку спрайта. Числа могут быть десятичными, шестнадцатеричными (перед каждым должен стоять знак $) или двоичными (перед каждым должен стоять знак BIN).

Вывод спрайтов и их объединение в группы (динамические спрайты) выполняется отмеченными выше командами. В состав расширения Sprite-BASIC входят две вспомогательные программы. Одна из них SP DEMO содержит подробную инструкцию по применению указанных выше (и некоторых других) команд языка и демонстрационную программу – игру "Космическая война", суть которой заключается в отражении лазером атаки космических объектов (причудливых облаков, космических кораблей и т.д.). В связи с этим нет необходимости подробно описывать действие команд этой версии. Отметим лишь, что скорость генерации спрайтов выше 50 образов в секунду, что позволяет получать достаточно быстрые и плавные движения до 6–10 объектов одновременно.

Вторая вспомогательная программа SP EDITOR (спрайт-редактор) резко облегчает создание пользователем собственных библиотек спрайтов. При ее вводе выводится сетка из 16×16 квадратов, в которой можно быстро перемещать курсор в любом направлении. Нажатие клавиши P затемняет квадрат, а W делает его светлым. Так в матрицу можно занести любую фигуру. В другой части экрана выводятся в естественном виде 32 знакоместа спрайта. Редактор позволяет вращать спрайты, перемещать их, объединять в группы, записывать на магнитофон и считывать их с него. Таким образом, редактор полностью берет на себя функции программирования операций по заданию спрайтов и объединению их в группы – динамические спрайты. Пользователю остается лишь выполнить художественную часть работы – изобразить спрайты.

5.10. ЛОГОГРАФИКА (ОПЕРАТОРЫ RENDOWN, PENUP, MOVE, TURN, TURN TO)

Машинная графика на языке Бейсик базируется на концепции построения геометрических фигур по заданным координатам их характерных точек. Для ряда геометрических построений это неудобно. Так, движение ручки или кисти человеческой руки, вращение сложных фигур и другие операции неудобно выполнять, анализируя каждый раз координаты характерных точек изображаемого объекта.

Альтернативным подходом к построению графиков является концепция рисования движущейся черепашкой (turtle). Имеется в виду, что че-

репашка есть некий воображаемый объект, несущий перо. Оно может быть поднято (тогда перемещения черепашки не видны) или опущено (перемещения сопровождаются построением линий). Черепашка может поворачиваться на заданный угол (в градусах) вправо или влево и двигаться вперед или назад на заданное число шагов (пикселей). Движения черепашки может повторять робот, управляемый ПЭВМ.

Концепция черепашки нашла наиболее полное отражение в языке программирования Лого (LOGO). Этот язык в последнее время получил широкую известность как язык, весьма удобный для обучения основам структурного программирования и машинной графики. Концепция черепашки оказалась столь удачной, что отдельные операторы Лого стали вводить в другие языки программирования для ПЭВМ: Бейсик, Паскаль, микроПролог и др.

Основная задача логографики заключается в перемещении точки-черепашки из положения с текущими координатами XCOR и YCOR вдоль направляющей вектора с углом φ = HD (heading) на расстояние L (команда FORWARD или FD). Для этого вычисляются новые координаты

XNEW = XCOR + MX * L * COS (HD),
YNEW = YCOR + MY * L * SIN (HD),

где MX и MY – масштабирующие множители.

Если световое перо опущено, на это указывает значение переменной статуса светового пера PS (pen status), то точки (XCOR, YCOR) и (XNEW, YNEW) соединяются отрезком прямой с помощью операторов DRAW и LINE. Если иначе, то отрезок не строится. Угол φ задается командой SET HD φ или TURN TO φ, увеличение φ на $\Delta\varphi$ обеспечивает команда LEFT $\Delta\varphi$ или LT $\Delta\varphi$ (поворот влево), уменьшение φ на $\Delta\varphi$ – команда RIGHT $\Delta\varphi$ или RT $\Delta\varphi$ (поворот вправо). После построения очередного отрезка задается XCOR = XNEW и YCOR = YNEW. С помощью множителей MX и MY можно менять масштаб изображения по осям X и Y (например, превратить окружность в эллипс).

Версия QL SUPERBASIC содержит пять основных операторов логографики [52]:

PENDOWN – перо вниз,
PENUP – перо вверх,
MOVE n – передвинуть черепашку на n шагов (вперед при $n > 0$ и назад при $n < 0$),
TURN φ – повернуть черепашку на угол φ (в градусах, направо или налево в зависимости от знака φ),
TURNTO φ – установить угол поворота черепашки φ.

Отличие операторов TURN φ и TURNTO φ состоит в том, что TURN φ поворачивает "тело" черепашки на угол φ относительно текущего его положения, а TURNTO φ устанавливает "тело" черепашки на угол φ относительно исходного положения ($\varphi_0 = 0$).

Язык Лого содержит в несколько раз больше команд управления черепашкой. Однако в Бейсике имеются аналоги отсутствующих команд.

```
10 CLS  POINT 120,60. PENDOWN
20 FOR J=1 TO 6
30 TURNTO J*60
40 FOR I=1 TO 6
50 MOVE 40  TURN 60
60 NEXT I  NEXT J
```

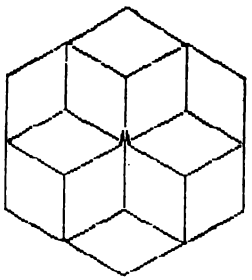


Рис. 5.25

Использование логографики на Бейсике иллюстрирует программа на рис. 5.25. Экран очищается, черепашка помещается в исходную точку (120,80) и опускается ее световое перо (строка 10). Затем 6 раз повторяются следующие операции: поворот черепашки командой TURNTO на 60° и шестикратное повторение продвижения вперед на 40 шагов с поворотом командой TURN на 60°. В результате вращения исходного шестиугольника относительно исходного угла строится 6 шестиугольников. Этот пример наглядно иллюстрирует простоту построения фигур вращения с помощью операторов логографики.

Следует отметить, что многие операторы языка Лого отличны от подобных операторов Бейсика с логографикой. Так, в Лого отсутствуют операторы MOVE, TURN, TURNTO, вместо них применяются FORWARD – вперед, BACK – назад, HEADETURTLE – повернуть черепашку и т. д. Поэтому прямой перевод программ с языка Лого на Бейсик затруднен. Речь идет о дублировании функциональных возможностей логографики. Следует также отметить, что обработка данных на языке Лого коренным образом отличается от принятой в Бейсике и всецело базируется на концепциях структурного программирования и оперирования данными в виде слов и списков (без деления данных на числовые и символьные).

5.11. ЗАДАНИЕ ЗВУКОВЫХ ЭФФЕКТОВ (ОПЕРАТОРЫ BEEP SOUND, PLAY, SREP, SON, SOFF, *ZAP, *NOISE)

Большинство ПЭВМ могут создавать различные звуковые эффекты с помощью громкоговорителя, встроенного в саму ПЭВМ или дисплей-телевизор. Эти эффекты могут быть весьма разнообразными: простые звуковые сигналы, музыкальные звуки, звуки речи (особенно с синтезаторами речи) и специальные звуковые эффекты (щелчок клавиши, выстрел, раскаты грома, барабанная дробь и т. д.).

Создание звуковых сигналов. Для создания звукового сигнала заданной длительности и тона служат операторы

[НС] BEEP *p, t*
[НС] SOUND *t, p*

Так, выполнение строки

```
10  FOR T=0 TO 100: BEEP . 5, T: NEXT T
```

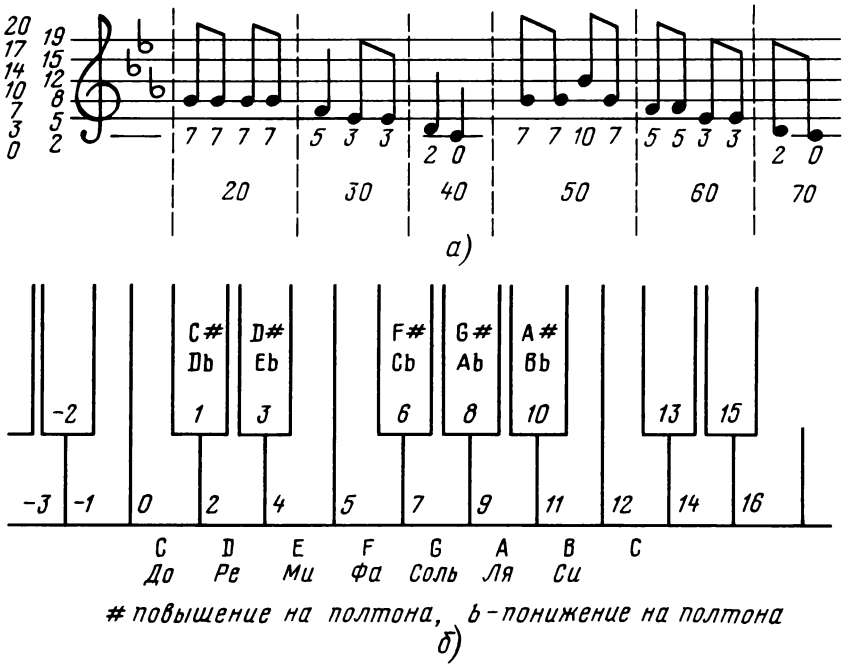


Рис. 5.26

создает множество звуковых тонов длительностью каждого 0,5 с. Тон постепенно повышается, и звук переходит в ультразвук, неслышимый ухом. Значения p могут задаваться от 0,01 или 0,02 до любого допустимого значения. Значения t могут быть и отрицательными (0 соответствует определенному тону). При $t>0$ частота звука растет, а при $t<0$ уменьшается.

Программа

```
1Ø FOR T=-6Ø TO 6Ø
2Ø BEEP . 5, T: NEXT T
```

позволяет воспроизвести весь спектр создаваемых ПЭВМ (ZX-Spectrum) тонов.

Соответствие звуковых тонов нотным знакам. Звуковые тона, формируемые с помощью операторов BEEP и SOUND, обычно однозначно соответствуют тонам, задаваемым с помощью нот (рис. 5.26,*а*). Таким образом, пользователь (музыкант или любитель) знакомый с нотной азбукой, может легко закодировать с нот понравившуюся ему мелодию. Соответствие тонов клавиатуры рояля (или другого музыкального инструмента) показано на рис. 5.26,*б*.

Приведенная ниже программа реализует исполнение фрагментов известной мелодии "Во поле береза стояла. . ." соответствует нотам на рис. 5.26,*а*:

```
  10 REM П.И.ЧАЙКОВСКИЙ "ВО ПОЛЕ
 БЕРЕЗА СТОЯЛА"
  15 INPUT "ЗАДАЙТЕ ТЕМП ";T: LE
T T=60/4/T
  20 FOR I=1 TO 4: BEEP T,7: NEX
T I
  30 BEEP 2*T,5: BEEP T,3: BEEP
T,3
  40 BEEP 2*T,2: BEEP 2*T,0
  50 BEEP T,7: BEEP T,7: BEEP T,
10: BEEP T,7
  60 BEEP T,5: BEEP T,5: BEEP T,
3: BEEP T,3
  70 BEEP 2*T,2: BEEP 2*T,0
  80 FOR I=1 TO 19: READ A: READ
 B: BEEP A,B: NEXT I
  90 DATA .25,7,.25,7,.25,7,.25,
7,.5,5,.25,3,.25,3
 100 DATA .5,2,.5,0,.25,7,.25,7,
.25,10,.25,7,.25,5,.25,5,.25,3,.
25,3,.5,2,.5,0
```

Звуковые тона могут задаваться операторами BEEP или SOUND двумя способами: с помощью аргументов (тон, длительность), входящих в эти операторы непосредственно и выбираемых из массивов данных. Оба эти способа реализованы в приведенной выше программе.

Мелодия задается с помощью операторов BEEP для каждой ноты (строки 20–70). При этом, меняя параметр T (число тактов) и параметр $t = 60/4T$ (t – четверть такта), можно менять и темп исполнения мелодии. Повторное исполнение мелодии задано с помощью одного оператора BEEP, аргументы которого с помощью цикла и оператора READ выбираются из предварительно составленного списка операторов DATA либо из иного массива (строки 80–100). Такой метод делает программы более универсальными и короткими.

Специальные звуковые эффекты. У большинства ПЭВМ музыкальные возможности ограничены одноголосым звучанием, полученным с помощью операторов задания звуков BEEP и SOUND. Однако у некоторых ПЭВМ (например, MSX и IBM PC) имеется возможность создания более сложных звуковых эффектов. Так, у ПЭВМ MSX команда BEEP (без аргумента) используется для создания короткого звукового сигнала, а функции оператора SOUND заметно расширены, он используется в виде

SOUND *M, N*

где *M* – номер (от 0 до 16) задающий характер звучания; *N* – байт. Задавая различные *M* и *N*, можно воспроизвести весь ряд музыкальных звуков по нотам.

Еще большими возможностями обладает оператор PLAY (играть), он обеспечивает трехголосное звучание ПЭВМ MSX. Оператор PLAY записывается в виде

PLAY СП1, СП2, СП3

где СП – строчная переменная. Строчные переменные для каждого из трех голосов записываются так же, как и в операторе DRAW этой ПЭВМ (в данном случае имеется в виду аналогия между понятиями "тянуть линию" и "тянуть звук"). В состав строчных переменных могут входить команды A*n*, B*n*, C*n*, D*n*, E*n*, F*n* и G*n* – задающие все семь основных нот (число *n*

задает такт $1/n$), Rn (пауза), знаки резкости # и +, знак бемоля и ряд других указаний. С последними пользователи данной ПЭВМ могут ознакомиться по инструкции по эксплуатации или с помощью книги [24]. К сожалению, перевод звуковых фрагментов программ с такими операторами на версии Бейсика большинства других ПЭВМ практически исключен.

В состав оператора PLAY ПЭВМ IBM PC могут входить указания даже о музыкальном ритме и стиле мелодии, например ритме танго, марша и т.д. У некоторых версий Бейсика число аргументов в операторах PLAY, SOUND и BEEP увеличено до 6–8, что позволяет создавать разнообразные звуковые эффекты.

Ряд особых команд управления звуком используется и в расширениях Бейсика. Например, в расширении Mega-BASIC имеются команды:
SREP_N – повтор звукового сигнала (SOUND REPEAT),
SON – включение звука,
SOF – выключение звука,
PLAY_n_1, n_2, n_3, n_4, n_5 – создание сложных звуковых сигналов.
Примеры их применения:

PLAY_Ø, 5, Ø, 3Ø, 5 – музыкальная трель,
SREP_1: SOUND_Ø, Ø, 7, 6, 25: SON – очередь из выстрелов,
SOUND_Ø, Ø, 7, 6, 25 – короткая очередь.

В расширении Sprait-BASIC имеются команды:
*ZAP – звук, напоминающий звучание спущенной тетевы лука,
*NOISE N – звук выстрелов (от одного при $N=1$ до очереди при $N=255$).

Имеются специализированные ПЭВМ – музыкальные синтезаторы. В опытных руках такой инструмент звучит как целый оркестр. Однако рассмотрение таких ПЭВМ особого применения выходит за рамки данной книги. Как уже отмечалось, в ПЭВМ внедряются синтезаторы речи, способные произносить слова и фразы (например, комментировать ходы игры в шахматы). Синтез звуков речи широко применяется и на программном уровне с помощью описанных выше команд. Это относится и к целым музыкальным произведениям.

Глава 6

УПРАВЛЕНИЕ ПЭВМ И ПЕРИФЕРИЙНЫМ ОБОРУДОВАНИЕМ, ЭКСПЛУАТАЦИЯ ИХ

6.1. УПРАВЛЕНИЕ ПЭВМ (ДИРЕКТИВЫ NEW, RUN, STOP, CONTINUE, END, CLR, CLRALL, VAC, SAC)

Современные ПЭВМ при включении приходят в готовность автоматически, что подтверждается сообщениями на экране ГОТОВ, READY, ОК и др. У некоторых ПЭВМ необходимо загрузить интерпретатор Бейсика с накопителя на магнитном диске (HMD) или кассетного магнитофона. Признаком готовности обычно является курсор – мигающая черточка, знак ␣ и т.д.

Перед вводом новой программы обычно используется директива NEW (новый). Она очищает ОЗУ и иногда придает новой программе заданное пользователем имя. Ее можно ввести в программу, например, для самоуничтожения последней после однократного исполнения. Это полезно при загрузке в ПЭВМ дополнительного алфавита.

Для пуска программ служит директива RUN (бег). Эта директива очищает ОЗУ, снимает определения переменных (или обнуляет их), устанавливает на нуль служебные счетчики циклов, подпрограмм и стека данных. Команда RUN НС запускает программу со строки НС, а команда RUN⟨ИМЯ⟩ (не у всех ПЭВМ) – программу с указанием имени, в этом случае выводится имя программы; команда RUNNH запускает программу без выдачи имени.

Для останова программы используются команды STOP и END. Команда STOP обеспечивает останов в том месте, где указано (например, ОСТАНОВ В СТРОКЕ 1ØØ). Команда END обычно помещается в конце программы. В ряде версий Бейсика этой команды нет, и останов происходит автоматически после выполнения строки с наибольшим номером. Обычно команда END закрывает все файлы (см. далее) и свидетельствует о завершении записи программы на магнитные носители. Все подпрограммы должны находиться выше команды END.

Встречаются и дополнительные директивы управления: CLR *n* (clear – стирание) – стирание программы из блока *n*; CLR ALL – стирание всех программ; CLEAR *a* – очистка ОЗУ с адреса *a*; VAC (Variable All Clear) – стирание всех переменных; SAC (Statistic All Clear) – стирание переменных статистики; C – стирание последнего символа; AC – полное стирание; CONT или CONTINUE – продолжение выполнения программы после останова по оператору STOP; ON – включение ПЭВМ.

При отладке программ увлеченный пользователь может нередко забыть о времени. Полезно, чтобы ПЭВМ постоянно напоминала ему о времени, индицируя его или подавая звуковой сигнал. У многих ПЭВМ есть встроенные кварцевые часы, и можно время индицировать, подав команду CLOCK (или подобную ей). При этом на экране дисплея индицируется текущее время, например, в виде 00:12:37, что означает 00 часов 12 минут и 37 секунд. Время отсчитывается от момента включения ПЭВМ.

Команда CLOCK в более общем виде

CLOCK [A] [ЧЧ ММ СС]

обеспечивает ряд операций. Если модификатор включения сигнала A (от слова alarm – сигнал) отсутствует, то команда CLOCK ЧЧ ММ СС (где ЧЧ – часы, ММ – минуты и СС – секунды) задает начальное время отсчета. Если модификатор A введен, то команда CLOCK ЧЧ ММ СС задает момент времени, достижение которого фиксируется ПЭВМ каким-либо действием, заданным отдельной командой CLOCK *N*, где *N* – число (код). В зависимости от *N* в момент, когда время достигает установленного значения, ПЭВМ может включить звуковой или особый визуальный сигнал,

вывести табло часов на экран дисплея, обеспечить переход к выполнению заданной подпрограммы. Например, CLOCK 1 выводит табло часов CLOCK Ø устраняет его с экрана дисплея.

В любой момент времени показания часов доступны пользователю в виде символьного значения специальной функции

TIME$

Удобно придавать значение функции TIME$ символьным переменным и выдавать его на печать (в форме ЧЧ : ММ : СС) в любой удобный момент времени.

Слудет отметить, что наряду с возможностью контроля времени в ходе оперативной работы на ПЭВМ, команда CLOCK и функция TIME$ имеют важное самостоятельное значение. Они, например, очень полезны для создания обучающих программ, когда необходимо задавать учащемуся определенное время на решение какой-либо задачи. Нужны эти команды и в том случае, когда ПЭВМ управляет различными объектами: роботами, станками, бытовыми приборами и др.

6.2. ВВОД ПРОГРАММЫ, ВЫВОД ЛИСТИНГА, РЕДАКТИРОВАНИЕ И ТРАССИРОВКА (ДИРЕКТИВЫ HOME, LIST, LLIST, EDIT, CLEAR, DELETE, INC, AUTO, RENUM, TRACE, CLS, SCR, WIDTH)

Ввод программы. Для ввода программы обычно не требуется каких-либо специальных команд. Набрав номер строки, вводят команды (операторы, функции и директивы). Пока ввод строки не фиксирован, действует система редактирования строки. В простейшем случае ее действие сводится к стиранию ошибочно введенного символа нажатием клавиши забоя (ЗБ, С и т.д.). После этого вводится нужный символ. Например (знаком_отмечен курсор):

```
Введено              1Ø INPUT Y_
Нажата клавиша ЗБ    1Ø INPUT _
Нажата клавиша X     1Ø INPUT X_
```

В этом примере с помощью клавиши ЗБ последняя ошибочно введенная буква Y стерта и исправлена на нужную X.

Редактирование программ. Современные ПЭВМ имеют более гибкие средства редактирования. Оно обычно выполняется с помощью курсора – специального знака, перемещаемого накопителем клавиш ← и →, а также устанавливаемого в исходное состояние вводом директивы HOME (домой). В этом состоянии курсор находится и после включения ПЭВМ. Так, замена ошибочного знака в середине строки достигается подводом под этот знак курсора и вводом нового знака, например:

```
Введено                  1Ø INPUT X_
Нажата и задержана
клавиша ←                1Ø INPUT X
Нажата клавиша P         1Ø INPUT X
```

Если клавиша ← или → нажимается дольше 1–2 с, то автоматически вводится режим автоповтора операции. Это значит, что курсор автоматически перемещается в соответствующем направлении с несколько большей скоростью, чем способен обеспечить пользователь, нажимая подряд клавишу ← или →.

Если необходимо вставить пропущенный знак, иногда используется директива INC (от слова include – включать). Она создает в строке пробел, на место которого вводится нужный знак.

Если место исправления находится в начале длинной строки (например, это номер строки), курсор можно сразу передвинуть в начало (вернуть "домой") с помощью директивы HOME, обычно вводимой нажатием специальной клавиши. Например:

Введено	5Ø INPUT X: PRINT EXP(X)_
Задана директива HOME	5Ø INPUT X: PRINT EXP(X)
Нажата клавиша 1	1Ø INPUT X: PRINT EXP(X)
Нажата клавиша EXE	1Ø INPUT X: PRINT EXP(X)

В этом примере номер строки 50 исправлен на номер 10. В конце нажатием клавиши EXE вводится отредактированная строка.

Авторедактирование в процессе ввода программы. У некоторых ПЭВМ (например, ZX-Spectrum) при вводе строк действует автоматическая система редактирования, предотвращающая принципиальные ошибки при наборе инструкций. Например, в этом случае невозможно набрать ошибочную инструкцию A = 1Ø: пропущен нужный в данной версии Бейсика оператор присваивания LET. При нажатии клавиши

A STOP
NEW

вместо буквы A окажется введенной команда NEW, в данном случае она абсурдна, но принципиально возможна. Столкнувшись с такой ситуацией, пользователь будет вынужден вспомнить, что перед предложением A = 1Ø должен стоять оператор LET. Набор правильной инструкции LET A = 1Ø произойдет уже без осложнений.

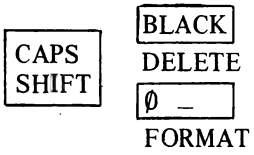
Нельзя ввести строку, содержащую синтаксическую ошибку, например:

```
1Ø PLOT 1ØØ,5Ø,4Ø[L]
```

Здесь ошибочно введены последние знаки (,4Ø); знаком [L] обозначен мигающий курсор. Если нажать клавишу ввода строки ENTER, такая строка не будет введена, и в служебной зоне появится мигающий знак вопроса ? перед первым ошибочным знаком:

```
1Ø PLOT 1ØØ,5Ø?,4Ø
```

В данной ПЭВМ для уничтожения знака перед курсором надо нажать клавиши



– вводится директива DELETE (уничтожить). Повторив эту операцию 3 раза, можно уничтожить знаки, 4 и Ø. После этого нажатием клавиши ENTER вводится отредактированная строка

```
1Ø PLOT 1ØØ, 5Ø
```

которая переходит из нижней части экрана (служебной зоны) в верхнюю (рабочую зону). Это свидетельствует о том, что ПЭВМ приняла строку.

Вывод листинга программы. В процессе ввода программы на экране дисплея появляется листинг – текст программы. Однако в этом случае он неупорядочен: содержит строки в произвольном, заданном пользователем порядке, строки, которые пользователь затем изменил (только некоторые ПЭВМ исключают из листинга измененные пользователем строки) и т.д.

Для вывода упорядоченного листинга программы служат специальные директивы:

LIST – вывод листинга начиная со строки с наименьшим номером,

LIST НС – вывод листинга начиная со строки с номером НС,

LIST НС1, НС2 – вывод листинга начиная со строки с номером НС1 и кончая строкой с номером НС2.

В такой форме листинг обычно выводится на экране дисплея. У ПЭВМ, работающих с принтером, имеются модификации этих команд. Например, у системы подготовки программ на базе ЭВМ "Электроника Д3-28" после слова LIST вводится знаки #1; если нужно вернуться вновь к выводу листинга на экран дисплея, после слова LIST вводится знак #Ø. У ПЭВМ ZX-Spectrum директива LIST выводит листинг на экран дисплея, а директива LLIST – на принтер.

Редактирование введенной программы. У многих ПЭВМ имеется специальная директива EDIT (редактирование), позволяющая вывести заданную строку в служебную область экрана дисплея. Далее к ней применимы все описанные выше приемы редактирования. У некоторых ПЭВМ используется директива EDIT в виде

```
EDIT НС
```

Она выводит в служебную зону строку с номером НС. У ПЭВМ ZX-Spectrum после номера строки появляется знак >. Нажимая клавишу $\frac{\text{CAPS}}{\text{SHIFT}}$ и клавиши ↓ и ↑, можно перемещать знак > вниз или вверх, устанавливая его на нужную строку. После этого нажатием клавиши EDIT указанная строка выводится в служебную зону.

Стирание строк программы. В ходе отладки программ бывает необходимо стереть или добавить строки. Для стирания одной ненужной строки дос-

таточно указать ее номер НС и нажать клавишу перевода строки (ПС, ВК, ENTER, EXE, RETURN, ↵ и т.д.).

Стирание одной строки НС1 или группы строк с номерами от НС1 до НС2 выполняется с помощью директив CLEAR (стирание) или DELETE (уничтожение):

CLEAR НС1 [, НС2]
DELETE НС1 [, НС2]

Например, директива CLEAR 1ØØ стирает строку 1ØØ, директива CLEAR 1ØØ, 2ØØ – все строки с номерами от 100 до 200.

Директивы CLEAR и DELETE у некоторых ПЭВМ могут иметь и другой смысл. Например, у ПЭВМ IBM PC директива CLEAR означает стирание всех данных с сохранением в ОЗУ программы, у ПЭВМ ZX-Spectrum директива CLEAR ⟨Адрес⟩ используется для стирания содержимого в ОЗУ, начиная с указанного адреса.

Автоматическая нумерация строк. В некоторых версиях Бейсика (ПЭВМ "Искра-226", ДВК-2М и др.) имеется специальная директива автоматической нумерации строк

AUTO $[n_1, \Delta n]$,

где n_1 – номер первой строки; Δn – приращение строк. Если n_1 или Δn не указаны, по умолчанию они задаются равными 10. После ввода каждой строки номер следующей генерируется автоматически.

Вывод и определение положения курсора. В ряде версий Бейсика указание на нужное место листинга дается выводом на него курсора. Обычно для этого служат специальные клавиши перемещения курсора. Имеется также команда

LOCATE x, y, N

задающая положение курсора с координатами x и y (при $N=0$ курсор не виден, при $N\neq 0$ он виден). Функция CSRLIN вырабатывает номер строки, в которой находится курсор, а функция POS – номер столбца.

Очистка экрана дисплея. После редактирования программы и просмотра ее листинга часто желательно очистить экран дисплея для последующего вывода на него другой информации, например результатов вычислений, графиков, текстов.

Директива CLS вызывает полную очистку экрана, она весьма полезна и в программах, когда необходимо, например, менять рисунки или другую информацию по мере выполнения программы.

Применяются также несколько отличные формы описанных выше директив:

LIST – вывод листинга всей программы,

LIST LIN.НС – вывод листинга строки НС,

LIST – LIN.НС – вывод листинга части программы от начальной строки до строк НС,

LIST LIN.НС1 – LIN.НС2 – вывод листинга части программы начиная со строки НС1 и кончая строкой НС2,

LIST LIN.HC – вывод листинга части программы от строки НС до конечной.

При использовании этих директив выводится заголовок (название) программы. Если не нужно выводить заголовок программы, используется директива LISTNH, аналогичная директиве LIST.

Директива DELETE в некоторых версиях Бейсика записывается в виде

DEL [LIN.HC1 – LIN.HC2]

Ей действия в отношении строк с указанными номерами задаются так же, как директивы LIST.

Директива SCR ⟨Название⟩ обеспечивает стирание содержимого ОЗУ и присваивает новой программе, которая будет загружаться, заголовок (название).

Отметим также директиву WIDTH (ширина), которая задается в виде

[HC] WIDTH [N]

где N – число, определяющее максимально допустимое число знаков в строке. При отсутствии аргумента N автоматически устанавливается $N=80$. Директива WIDTH 40 означает, что максимальное число знаков в строке будет 40.

Перенумерация строк. После отладки программ строгий порядок нумерации строк часто нарушается: пользователь вставляет одни строки и удаляет другие. В расширенных версиях Бейсика используется специальная директива RENUM (от слова renumber – перенумеровать)

RENUM [n_1, m_1, Δm]

Здесь n_1 – номер первой строки программы, начиная с которой идет перенумерация; m_1 – новый номер первой строки; Δm – шаг строк в новой программе. В ходе перенумерации строк автоматически меняются адреса безусловных и условных переходов, а также адреса в командах RUN m, GOSUB m и ON. Редко, но встречаются случаи, когда перенумерация невозможна. В этом случае ПЭВМ выдает специальное указание. Перенумерация строк придает программам опрятный вид и облегчает их анализ.

Трассировка. Это особый вид отладки программ, вводится директивой TRACE ON (трассировка включена), а отключается вводом директивы TRACE OFF (трассировка выключена). При включении трассировки происходит выполнение программы отдельно по каждой команде с выводом листинга этой инструкции и выход на режим непосредственных вычислений. При этом переход от выполнения одной команды к другой происходит в соответствии с логикой выполнения программы, т.е. учитываются все безусловные и условные переходы, переходы к подпрограммам и возврат из них, выполнение тела циклов заданное число раз и т.д.

Для иллюстрации режима трассировки ПЭВМ FX-702P рассмотрим вычисление $y = \sin x/x$ при $x \neq 0$ и $y = \sin x/x = 1$ при $x = 0$ с помощью программы

```
1Ø  INP X
2Ø  IF X=Ø; PRT "Y=1"; GOTO 1Ø
```

```
30  PRT "Y="; SIN X/X
40  GOTO 10
```

Для ввода режима TRACE OFF задаем моду MODE 2. Последующие действия и показания дисплея даны ниже (программа в области P9):

Операция	Комментарий	Показания дисплея
F1 9	Пуск программы P9	P9 – 10 и затем ?
1 EXE	Ввод $x=1$	–
STOP	Останов счета	P9 – 10 INP X
CONT	Продолжение счета	P9 – 20
STOP	Останов счета	P9 – 20 IF X=0; PRT "Y="
CONT	Продолжение счета	Y=0.8414709848
STOP	Останов счета	P9 – 30 PRT "Y="; SIN X
CONT	Продолжение счета	–
STOP	Останов счета	P9 – 30 PRT "Y="; SIN X
CONT	Продолжение счета	P9 – 40
STOP	Останов счета	P9 – 40 GOTO 10
CONT	Продолжение счета	? (и т.д.)

Таким образом, манипулируя клавишами STOP и CONT, можно последовательно просмотреть начальный порядок выполнения инструкций. Оно прерывается выдачей результатов при выполнении оператора ввода INP и оператора PRT (сокращение от PRINT). При этом перед каждой инструкцией появляется номер программной области и номер строки, например, P9 – 10 означает, что выполняется инструкция для программной области P9 и строки в ней 10.

В версии Бейсика, описанной в [54], применяется более сложная система трассировки программ при их отладке. Здесь команда TRACE HC служит для обращения к специальной подпрограмме трассировки, которую пользователь должен составить сам (обычно ее размещают где-то в конце программы). Она помещается в любом месте основной программы и обеспечивает, начиная с этого места, ее трассировку. Команда TRACE 0 включает трассировку.

Ниже приведена небольшая программа, подвергаемая трассировке:

```
   1 REM DEMONSTRATIOON TRHCE
   5 TRACE 9999
  10 INPUT "X=";X
  20 PRINT "X=";X
     PRINT "EXP (X)=";EXP X
  30 GO TO 10
9999 PRINT INVERSE 1;LINO;":";st
at;
     LIST LINO-1 TO LINO
     RETURN
```

В системе трассировки предусмотрены две служебные функции: LINO – выдача текущего номера строки и STAT – выдача номера предложения в строке. Подпрограмма трассировки в строке 9999 выдает номер строки и номер предложения в ней, которые выполняются в данный момент. Результат выполнения программы с трассировкой выглядит так:

```
 10 INPUT "X=";X
 20 PRINT "X=";X
PRINT "EXP (X)=";EXP X
X=1
 20 PRINT "X=";X
PRINT "EXP (X)=";EXP X
EXP (X)=2.7182818
 30 GO TO 10
 10 INPUT "X=";X
 20 PRINT "X=";X
PRINT "EXP (X)=";EXP X
X=2
 20 PRINT "X=";X
PRINT "EXP (X)=";EXP X
EXP (X)=7.3890561
 30 GO TO 10
 10 INPUT "X=";X
```

Подобная организация трассировки на первый взгляд кажется сложной: нужно составить ее подпрограмму и ввести в программу команду TRACE HC. Зато в подпрограмму пользователь может ввести по своему усмотрению любые элементы трассировки, например вывод значений любых переменных, особые комментарии. В обычных условиях все эти элементы пришлось бы многократно вкрапливать в основную программу.

Выявление и устранение ошибок. Современные ПЭВМ ориентированы на диалоговый режим работы с пользователем. В частности, это предусматривает автоматический контроль ошибок, возникающих в ходе ввода и выполнения программ. Простейшие ПЭВМ дают знать об ошибках остановом работы и выдачей сообщений вроде:

ERR – 3 IN P9 – 2Ø

или

ОШИБКА 15 СТРОКА 1ØØ

Первое сообщение (ПЭВМ FX-702P) указывает на номер ошибки 2 (в данном случае некорректная операция), которая обнаружена в строке 20 программной области P9. Второе сообщение (комплекс СПП-2 на базе микроЭВМ "Электроника Д3-28") также указывает на номер ошибки (15) в строке 100. В более сложных ЭВМ указывается даже тип ошибки.

Если индицируется только номер (код) ошибки, нужно по таблице кодов идентифицировать ее. Такие таблицы имеются в технической документации на ПЭВМ. К сожалению, коды ошибок не стандартизированы. Количество кодов лежит от 10 до многих десятков. Наиболее распространенными являются такие ошибки (даны выборочно):

Sintaxis error – синтаксическая ошибка (например, апостроф вместо кавычек, точка с запятой вместо двоеточия и т. д.),

Invalid argument – неправильно задан аргумент (например, для символьной переменной задано числовое значение, аргумент функции выходит за пределы допустимого),

NEXT without FOR – нет заголовка цикла,

RETURN without GOSUB – оператор возврата из подпрограммы без обращения к ней,

GOSUB without RETURN – нет оператора возврата из подпрограммы.

Таким образом, ПЭВМ достаточно ясно подсказывает пользователю характер ошибки и место ее в программе.

После уточнения характера ошибок нужно вывести листинг участка программы с ошибочной строкой или саму эту строку (директива LIST), внимательно просмотреть строку с ошибкой. Чаще всего ошибка, распознанная ранее, легко выявляется. Бывают более сложные случаи, например, когда ПЭВМ указывает на ошибку в задании аргумента или некорректную математическую операцию. В таких случаях полезно вывести на печать (операторами PRINT или LPRINT) значения подозрительных переменных. Так, если появилось указание о некорректной математической операции в строке

```
15Ø  LET C=B/A
```

то подозрение падает на переменную A (если A = 0, то возникает указанная ошибка). Проверить это можно, дав команду PRINT A и наблюдая выведенное значение данной переменной.

В ряде версий Бейсика имеются специальные средства редактирования и отладки программ [54], например директивы:

EDIT ⟨Имя⟩ – вызов на редактирование переменных или процедур с указанным именем;

LIST ⟨Имя⟩ – вызов листинга поименованных процедур;

LIST FORMAT ⟨Номер⟩ – переключение формата вывода листингов (зависит от номера $N = 0 - 5$, можно, например, выводить листинг без номеров строк, листинг структурированных программ с одним или двумя отступами);

LIST PROC ⟨Имя⟩ – вывод листинга указанной процедуры;

LIST REF – выдача номеров всех строк, в которых встречается определенная переменная;

DELETE ⟨Имя⟩ – стирание переменных, массивов или процедур с указанным именем;

JOIN ⟨Номер строки⟩ – объединение указанной строки со следующей.

В некоторых случаях необходимо, чтобы программа продолжала работу и при возникновении ошибки, перейдя, например, к альтернативному варианту расчета. Для этого можно использовать команды:

```
ON ERROR GOTO НС
```

или

```
ON ERROR GOSUB НС
```

– если возникает ошибка, то осуществляется переход к строке с номером НС или обращение к подпрограмме, начинающейся со строки НС.

Специальные отладочные средства. Для ПЭВМ разработан ряд специальных сервисных программ-утилитов (от слова utility – полезный) для отладки и редактирования программ пользователя на языке Бейсик. Широкое распространение получили, например, программы TOOLKIT, ULTRAKIT, включающие в Бейсик указанные выше средства отладки программ, если их нет в базовой версии.

Весьма полезной директивой отладочных программ является директива компрессии (COMPRESSE) – объединения всех строк в одну, там, где это допустимо структурой программы. При этом автоматически меняются номера строк условных и безусловных переходов, подпрограмм и т.д. В результате резко уменьшается общее число номеров строк и команд их окончания, сокращается листинг программы, уменьшается емкость ОЗУ, занимаемая программой и несколько уменьшается время выполнения программы. Однако листинг программы после компрессии становится наглядным. Нередко пользователь с удивлением видит, что его программа, содержащая десятки коротких строк, компенсируется в 2–3 строки, каждая из которых занимает чуть не весь экран дисплея.

Компиляция бейсик-программ. Программы на Бейсике выполняются сравнительно медленно (различие в сравнении с операциями на машинных кодах или ассемблере нередко достигает сотен раз). Поэтому в последнее время наряду с интерпретирующими версиями языка Бейсик получили распространение компилирующие версии. Напомним, что компиляция есть автоматический перевод всей бейсик-программы на язык машинных кодов. Скомпилированная программа обычно выполняется намного быстрее, чем обычная. Так, выигрыш при использовании целочисленных компиляторов доходит до 50–100 раз, а полных до 2–5 раз. Однако следует учитывать, что например, целочисленные компиляторы не могут работать со многими функциями: EXP, LN, SIN и др.

Пуск скомпилированных программ производится директивами CALL ⟨Адрес⟩, USR ⟨Адрес⟩, RANDOMIZE USR ⟨Адрес⟩ и т.д. (зависят от типа ПЭВМ). Исходная бейсик-программа может быть стерта (рекомендуется все же хранить ее копию на всякий случай). Скомпилированные программы выполняются при предварительно загруженном в ОЗУ компиляторе.

Оптимизация бейсик-программ. Оптимизация может производиться по различным и нередко противоречивым критериям, например сокращению листинга программы и занимаемой ею емкости ОЗУ, повышению скорости счета, улучшению наглядности. Сейчас предпочтение отдается наглядности программ, и в них рекомендуют включать подробные комментарии. Однако, следуя правилу "меньше слов – больше дела", не следует впадать в крайности и в этом.

В ряде случаев необходимо повышение эффективности программ, прежде всего уменьшение времени счета. Для этого нужно оценить реальное быстродействие ПЭВМ, вычислив время одного пустого цикла FOR – NEXT Вставив в цикл интересующую нас арифметическую операцию, можно найти время ее выполнения (для этого цикл повторяется 100 или 1000 раз). Обычно для выполнения арифметических операций требуется от 3–4 до 30–40 мс, а для функции SIN, COS, LN, EXP и др. – примерно на порядок больше.

Правила повышения эффективности расчетных программ описаны в [1]. Отметим важнейшие из них.

Прежде всего нужно помнить о возможности преобразования расчетных выражений к виду, обеспечиваемому повышение скорости счета на данной

ПЭВМ. Так, рекомендуется тщательно "чистить" циклы, вынося за их пределы все повторяющиеся вычисления. Например, из двух формул:

$$S = \sum_{i=1}^{100} (1 - e^{-2})i \quad \text{и} \quad S = (1 - e^{-2}) \sum_{i=1}^{100} i$$

следует пользоваться правой, ибо в ней член $(1 - e^{-2})$ вычисляется один раз, тогда как в левой 100 раз.

Вместо вычисления x^2, x^3 и т.д. следует вычислять $x \cdot x$, $x \cdot x \cdot x$ и т.д., поскольку операция возведения в степень x^n одна из самых медленных. Рекомендуется не применять повторяющиеся арифметические операции и в других случаях. Так, вместо вычисления

```
LET A(2*I-1)=A(2*I-1)*A(2*I-1)+C
```

следует вычислять

```
LET N=2*I-1: LET A(N)=A(N)*A(N)+C
```

Следует учитывать, что вызов простых переменных происходит быстрее, чем индексированных, что операции вывода на печать требуют заметного времени (надо выводить только нужные данные и т.д.).

Кардинальным путем повышения эффективности является правильный выбор алгоритма с учетом возможностей конкретных ПЭВМ (см. пример в § 3.1, когда выигрыш был достигнут за счет использования матричных операций, выполняемых ПЭВМ быстро).

6.3. УПРАВЛЕНИЕ КАССЕТНЫМ НАКОПИТЕЛЕМ (ДИРЕКТИВЫ SAVE, PUT, LOAD, GET, MERGE, VERIFY, MOTOR, SCREEN$)

Кассетный накопитель предназначен для записи и считывания программ и данных.

У большинства ПЭВМ предусмотрена возможность применения в качестве кассетного накопителя обычного бытового кассетного магнитофона. При выборе кассетного накопителя и кассет для него следует учитывать способ записи на магнитную ленту. Чаще всего у ПЭВМ при записи используются двухтоновые сигналы и запись идет с обычным высокочастотным подмагничиванием. В этом случае не рекомендуется применять специальные кассеты для цифровой записи (Digital Kassette), поскольку они рассчитаны на иной способ записи без высокочастотного подмагничивания (последние используются в профессиональных накопителях).

На кассету МК-90 (время звучания 90 мин при стандартной скорости протяжки ленты 4,76 см/с) можно поместить информацию объемом 0,8 – 1,2 Мбайт. Ее сохранность выше, чем на магнитных дисках, поэтому и рекомендуется дублировать программы на ленте, даже если имеется дисковый накопитель. Главный недостаток кассетного накопителя – большое время поиска и загрузки (программы объемом 40–45 Кбайт загружаются обычно 3–4 мин).

Запись программ и данных. Для записи (хранения) программ обычно используется директива

[НС1] SAVE [Название] [LINE [НС2]]

обеспечивает присвоение ей названия и указывает номер строки НС2, с которой начнется выполнение программы после загрузки.

У некоторых ПЭВМ одновременно с программой записываются и данные. Директива

[НС] SAVE ["Название"] DATA ⟨Переменная⟩ ( )

обеспечивает запись на магнитофон массива данных с присвоением ему названия. Например, строка программы

```
1ØØ SAVE "МАССИВ" DATA Y ( )
```

записывает массив переменной Y с именем "МАССИВ". При записи отмечается размерность и тип массива. Директива

[НС] SAVE ["Название"] CCODE A1 [, A2]

записывает содержимое ОЗУ в виде кодов, начиная с адреса A1 и кончая адресом A2, с присвоением блоку кодов названия. Если адрес A2 не указан, называется содержимое всего ОЗУ, начиная с адреса A1.

У ПЭВМ FX-702P, имеющих программные блоки, директива записывается в виде

[НС] SAVE [#*n* "Название"]

– запись программы в блок P_n и присвоение ей названия (до 8 одинаковых символов, например A, BBB, * * * * и т. д.),

[НС] SAVE ALL ["Название"]

– запись подряд всех программ (от блока P0 до P9).

Для записи данных у ПЭВМ FX-702P используется директива PUT (помещать) в виде

[НС] PUT ["Название"] {Список переменных}

Переменные в списке должны указываться строго по порядку: от A до Z, от A(Ø) до A(199). Например, директива PUT "D" A, C, D записывает блок данных с именем D, содержащий значения переменных A, C, D. Записывать его в виде PUT "D" D, C, A нельзя (ПЭВМ укажет ошибку). Если надо записать, к примеру, значения всех переменных от A до F и значения переменных массива от A(2Ø) до A(1ØØ), то используется директива PUT в виде

PUT "D" A, F, A(2Ø), A(1ØØ)

В системах подготовки программ на базе ЭВМ "Электроника Д3-28" применяются следующие виды команд:

SAVE ['Название'] [НС1, НС2] – запись программы начиная со строки НС1 и кончая строкой НС2 с присвоением ей названия;

SAVE END – служебная запись, указывающая на конец ленты;

DATA SAVE ['Название'] {Список данных} – запись блока данных (констант, значений переменных и массивов, перечисленных в списке данных и разделенных запятыми).

Например:

SAVE 'СТАРТ' 3Ø, 15Ø

записывает часть программы со строки 30 до строки 150 с именем СТАРТ, DATA SAVE 'ДАННЫЕ' #PI, A, B, C, X(5), Y(5,5) – блок данных с именем ДАННЫЕ, содержащий константу π, значения переменных A, B и C, одномерный A(5) и двухмерный Y(5,5) массивы.

Считывание программ и данных с кассетного магнитофона и их загрузка в ОЗУ. Для считывания программы с магнитной ленты магнитофона и загрузки ее в ОЗУ служит директива LOAD. Обычно ее ввод ведет к стиранию старой программы и данных. Директива LOAD используется в виде

LOAD ["Название"]

– вводит в ОЗУ программу (иногда вместе с данными), считываемую с магнитной ленты магнитофоном,

LOAD ["Название"] DATA ⟨Переменная⟩ ( )

– вводит в ОЗУ данные – значения переменных или массивов,

LOAD ["Название"] CODE A1 [, A2]

– вводит в ОЗУ коды, размещая их начиная с адреса A1 и кончая адресом A2.

Запись и считывание изображений. Для записи на магнитную ленту массива данных, создающих полное изображение на экране дисплея, а также для считывания этих данных и их загрузки в ОЗУ (восстановления изображения на экране) совместно с директивами SAVE и LOAD используется модификатор SCREEN$ (экран) [26]. Этот модификатор используется в виде

[HC] SAVE ["Название"] SCREEN$

– записывает изображение (в виде данных) и присваивает ему название,

[HC] LOAD ["Название"] SCREEN$

– считываемое с ленты изображение заносится в ОЗУ ПЭВМ, что ведет к его появлению на экране дисплея.

Директива LOAD у ПЭВМ FX-702P используется в виде:

[HC] LOAD [#*n* "Название"]

– загружает в ОЗУ программу из блока P*n* с указанным названием,

[HC] LOAD ALL ["Название"]

– загружает в ОЗУ все программы (из блоков P0 – P9) и данные.

Для считывания только данных и их загрузки в ОЗУ у этой ПЭВМ служит команда GET (получать):

[HC] GET ["Название"] {Список переменных}

Список переменных составляется по правилам, приведенным при описании директивы PUT. Например, при выполнении строки

1Ø GET "D" A, Z

в ОЗУ записываются значения переменных от A до Z из файла данных с именем (названием).

У ПЭВМ FX-702P выполнение директивы LOAD не стирает старую программу. Новая программа соединяется со старой (т.е. строки новой программы замещают строки старой), если их номера совпадают. Если номера не совпадают, то строки старой программы сохраняются.

У большинства ПЭВМ загрузка новой программы с помощью оператора LOAD ведет к уничтожению старой программы и относящихся к ней данных. Для объединения новых частей программ и данных со старыми служит специальная директива MERGE (соединять). Правила ее использования аналогичны описанным для директивы LOAD. Аналогично директиве MERGE иногда используется директива APPEND (присоединять).

Индикация файлов. У большинства ПЭВМ предусмотрена индикация на экране дисплея типа и наименования блоков информации – файлов (см. подробнее в следующем разделе), считываемых в данный момент с магнитофона. Например, у ПЭВМ FX-702P считывание программного файла с именем D сопровождается появлением сообщения

PF : D

где PF означает программный файл. Если была задана директива LOAD #5 "D", то после считывания появится сообщение

READY P5

Оно свидетельствует о том, что этот файл размещен в программной области P5, после чего можно запускать программу, имеющуюся в этой области.

Считывание файла данных с символами AAA сопровождается сообщением

VF: AAA

где VF (от слов variable file) указывает на считывание файла данных (переменных). Соответственно считывание объединенного файла всех программ и данных (команда LOAD ALL) ведет к появлению сообщения

PVF: AAA

У более сложных ПЭВМ имя (название) программ при использовании директив SAVE, LOAD и MERGE может состоять из 8–10 и более любых символов. При считывании (директивы LOAD и MERGE) на экран дисплея выводится список считанных файлов, например Program Start – программа с именем Start, Byte B – массив данных в виде байтов с именем B, Code C – массив кодов с именем C, Program Finish – программа с именем Finish и т.д.

К сожалению, у некоторых моделей микроЭВМ (например, у систем подготовки программ на базе ЭВМ "Электроника Д3-28") тип и название файла в ходе считывания не индицируются. Это затрудняет поиск файлов на кассете, особенно при использовании ускоренной перемотки ленты, когда нужный файл легко пропустить.

При аккуратном обращении с магнитофоном и лентой надежность записи и считывания файлов весьма высока (возможно появление одной ошибки на $10^7 - 10^9$ команд). Однако при использовании старой ленты или разрегулировании лентопротяжного механизма надежность записи резко снижается. Поэтому у большинства ПЭВМ предусмотрены специальные меры по контролю качества записи и считывания.

Проверка программ после записи на кассетный магнитофон. Для этой цели у многих ПЭВМ используется директива VER или VERIFY (проверять), которая применяется аналогично директиве LOAD, но сразу после записи программ и данных командой SAVE. Директива VERIFY обеспечивает сравнение считанного с помощью магнитофона файла с тем файлом, который использовался на записи. Если файлы идентичны, ПЭВМ выходит на режим готовности выполнения программ. Если обнаружено расхождение, на экран дисплея выводится сообщение о сбое в считывании файла. В этом случае надо повторить запись, используя исправные кассету и магнитофон. Следует отметить, что директива VERIFY позволяет проверить лишь качество записи в момент ее проведения. Порча записи в последующем ведет к непригодности программ.

У ПЭВМ Aquarius запись и считывание программ осуществляются директивами

```
CSAVE     "Имя программы"
CLOAD     "Имя программы"
```

Имя программы должно состоять не более чем из 6 знаков.

Проверка правильности записанной программы осуществляется директивой

```
CLOAD ? "Имя программы"
```

Соединение программы, считываемой с магнитофона, с программой, хранящейся в ОЗУ этой ПЭВМ, не предусмотрена.

Для записи и считывания данных используются директивы

```
CSAVE *⟨Имя переменной или массива⟩
CLOAD *⟨Имя переменной или массива⟩
```

При использовании массивов их нужно предварительно определить с помощью оператора DIM. Например, в программе

```
1Ø    DIM X(5Ø)
.................
1ØØØ  CSAVE *X
```

в строке 10 задан массив X(50) из 51 элемента (нумерация с 0). После выполнения программы (многоточие) выполнение строки 1000 обеспечит запись массива X(50) на магнитофонную ленту. При считывании в программе

```
1Ø DIM X(5Ø)
2Ø CLOAD *X
```

также резервируется память под массив (строка 10), а затем задается считывание его с ленты.

Автоматическое включение и выключение магнитофона. Интерфейсы ряда ПЭВМ (например, FA-2 ПЭВМ FX-702P) имеют устройство для автоматического включения двигателя магнитофона и его выключения в ходе записи и считывания программ. Это позволяет организовать программно-управляемый обмен информацией между ПЭВМ и периферийным оборудованием. Для этого составляется специальная (обычно короткая) программа, которая последовательно вызывает файлы программ и данных, загружает их в ОЗУ и обеспечивает выполнение необходимых операций. При этом суммарный объем данных и программ может быть намного больше того, который характерен для ОЗУ ПЭВМ.

У некоторых ПЭВМ (например, IBM PC) для включения двигателя магнитофона используется директива MOTOR 1, а для выключения MOTOR 0. Ее также можно использовать для организации программно-управляемого обмена информацией. Если директива MOTOR используется без кода (0 или 1), то при первом ее исполнении двигатель включается, при втором выключается, при третьем вновь включается и т.д., т.е. в этом случае директива MOTOR поочередно включает и выключает двигатель магнитофона.

6.4. УПРАВЛЕНИЕ НАКОПИТЕЛЕМ НА МАГНИТНЫХ ДИСКАХ (ДИРЕКТИВЫ DIR, CATALOG, CAT, UNIT, RENAME, DELETE, LOAD, SAVE, KILL И ДР.)

Накопитель на гибких магнитных дисках (НГМД) предоставляет пользователю существенно больше удобств, чем накопитель на магнитной ленте (НМЛ). Это связано, прежде всего, с малым временем доступа к информации, записываемой на диск. Так, у НГМД время переключения дорожек составляет доли секунды, такое же время считывания информации с одной дорожки или сектора. Небольшие программы считываются с диска за время от долей секунды до 1–2 с, и лишь большие программы могут считываться за 10–30 с (иногда и больше). В целом время доступа к программам и данным у НМД на 2–3 порядка меньше, чем при использовании НМЛ с лентой в стандартных кассетах, еще меньше при использовании накопителя на жестких дисках.

Информация записывается на диск и считывается с него в упорядоченной форме – в виде *файлов*. Она содержит заголовок (имя) и делится на отдельные блоки. В состав файла входит служебная информация, позволяющая определить вид файла (программный, двоичный или текстовый), указание о защите файла и т.д. Файлы могут храниться на магнитной ленте или дисках.

Файлы – весьма удобные средства для оперирования большими объемами информации самого разнообразного характера. Правильное владение техникой использования файлов свидетельствует о существенном повышении искусства и культуры программирования пользователем.

Современные ПЭВМ обычно используются с *дисковой операционной*

системой (ДОС) – комплексом специальных программ, значительно расширяющим возможности применения ПЭВМ и дискового накопителя*. При использовании ДОС информация (файлы программ и данных) автоматически размещается на диске. Файлы могут быть уничтожены (стерты), переименованы, дополнены, защищены от случайного стирания и записи и т. д. Для этого используется ряд специальных директив, описанных ниже.

Вывод каталога файлов. Для вывода перечня (каталога) всех файлов, имеющихся на НМД, служит директива DIR, CATALOG или CAT (каталог). В общем случае директива CATALOG вводится в виде

CATALOG [,*S* [,*D* [,*V*]]]

где *S* – номер разъема, к которому подключен контроллер накопителя; *D* – номер разъема на контроллере; *V* –,номер диска (он формируется при инициализации диска – см. ниже). Если номера в квадратных скобках опущены, то они автоматически устанавливаются относящимися к тому накопителю, в который вставлен диск. Далее указание [,*S* [,*D*[,*V*]]] будет обозначено сокращенно как [Y]. Таким образом, директива каталог будет записана в виде CATALOG [Y].

Запись файлов на диск, их разметка и уничтожение. Файлы, которые записываются на диск, можно подразделить на три типа (обозначения приняты как у ПЭВМ Apple-II и "Агат"): A – бейсик-программы, B – данные в двоичной форме и T – данные в текстовой форме. Для всех их используются следующие общие директивы:

INIT Имя [Y] – разметка диска и присвоение ему имени (разметка или инициализация диска заключается в создании на диске системы магнитных меток, которые используются при работе ДОС);

RENAME Имя 1, Имя 2 [Y] – переименование файла (здесь Имя 1 – старое имя файла, Имя 2 – новое имя файла);

DELETE Имя [Y] – уничтожение (стирание) файла с заданным именем;

LOCK Имя [Y] – защита файла на запись (такой файл в каталоге помечается знаком *; попытка осуществить в него запись ведет к сообщению о том, что файл защищен);

UNLOCK Имя [Y] – снятие защиты файла на запись;

MON [,*C*] [,*I*] [,∅] – включение отображения команд и информации на экране дисплея (*C* – задает вывод команд, *I* – текста, вводимого с накопителя, ∅ – текста, выводимого из ПЭВМ в накопитель, одно из этих указаний должно быть обязательно);

NOMON [,*C*] [,*I*] [,∅] – снятие (отключение) отображения, введенного командой MON;

MAXFILES *n* – резервирование *n* буферов для файлов размером 256 байт каждый (при начальной загрузке *n* = 3).

Для работы с файлами типа A используются следующие директивы:

SAVE Имя [Y] – запись программы на магнитный диск (если на диске была запись с таким именем она теряется);

---

* Вход в ДОС из Бейсика выполняется директивой SYSTEM (иногда BYE).

LOAD Имя [Y] – загрузка программы из диска в ПЭВМ (если в ней была программа и данные, они теряются);

RUN Имя [Y] – загрузка программы из диска в ПЭВМ и пуск ее (если в ПЭВМ была старая программа и данные, они теряются);

CHAIN Имя [Y] – загрузка программы из диска в ПЭВМ и запуск программы с передачей данных для новой программы (старая программа при этом теряется).

Работа с файлами. Для работы с *двоичными файлами,* характеризующимися начальным адресом A и длиной файла L, применяются команды:

BSAVE Имя A, L [Y] – запись файла на диск;

BLOAD Имя [, A [Y]] – считывание файла с диска и его загрузка в ОЗУ с адреса A (если он не указан загрузка идет с адреса, указанного при записи на диск);

BRUN Имя [Y] – операция, аналогичная предшествующей, но с передачей управления на начальный адрес файла.

Текстовые файлы могут быть двух типов: последовательного доступа, в которых хранится сплошная последовательность символов (эти файлы записываются или считываются от начала до конца), и прямого доступа, содержащие записи фиксированной длины. Для последнего файла можно указать, где выполняется запись или чтение. Далее обозначено: B – номер байта в файле с последовательным доступом или в записи файла с прямым доступом, R – номер записи в файле с прямым доступом и j – размер записи в файле с прямым доступом.

Открытие и закрытие файла. Прежде чем использовать файл, его нужно открыть. Для этого служит директива OPEN (открывать), присваивающая имя файлу:

[HC] OPEN Имя [Y] – для файла последовательного доступа,
[HC] OPEN L_j [Y] – для файла прямого доступа.

Открытие нового файла, т.е. ранее не существующего на диске, приводит к образованию пустого файла. Открытие файла с имеющимися уже на диске именами, позиционирует последний на начало.

Файл закрывается с помощью директивы CLOSE (закрывать):

[HC] CLOSE [Имя] – для файла последовательного доступа,
[HC] CLOSE Имя [Y] – для файла прямого доступа.

Директива CLOSE без указания имени закрывает все файлы.

Запись в файл и считывание из файла. Информация записывается в файл с помощью директивы WRITE (писать):

[HC] WRITE Имя [, B] – для файла последовательного доступа,
HC WRITE, Имя [, R, [, B]] – для файла прямого доступа.

Для наращивания (дополнения) файла последовательного доступа используется директива APPEND (присоединять):

[HC] APPEND Имя [Y]

открывающая файл и позиционирующая его на последний символ. После-

дующая директива WRITE будет наращивать файл. Защита файла от записи на файл, открытый директивой APPEND, не распространяется.

Информация считывается с файла с помощью директивы READ (читать):

[НС] READ Имя [,*B*] – для файла последовательного доступа,
[НС] READ Имя [,*B* [,*R*]] – для файла прямого доступа.

При считывании информация из файла загружается в ОЗУ ПЭВМ.

Исполнение из текстового файла. Команды, записываемые в текстовом файле, принимаются к исполнению с помощью директивы EXEC (от слова execute – выполнять):

[НС] EXEC Имя [,*R*] [Y]

После ее выполнения ввод команд из текстового файла выполняется с помощью программы-монитора до тех пор, пока не встретится команда закрытия файла CLOSE без предшествующей ей команды OPEN (либо до закрытия файла). Вводимые команды исполняются по мере их поступления, ввод запускаемых программ также производится из исполняемого файла (кроме случая, когда запущенная программа открыла для себя другой файл для чтения).

Позиционирование файла. Обеспечивает пропуск *R* записей в файле, осуществляется с помощью директивы POSITION (позиция):

[НС] POSITION Имя [,*R*]

У различных версий Бейсика эти директивы могут иметь несколько иную форму, например, могут содержать после слов директивы указание *#N*, где *N* – номер файла. Встречаются также несколько иные обозначения директив (версия Бейсик-80):

[НС] ATTRIB: F Номер дисковвода [,Имя] [W1]

– обеспечивает защиту файла от записи,

[НС] ATTRIB: F Номер дисковвода [,Имя] [WØ]

– обеспечивает запись, стирание и изменение имени ранее защищенного файла,

[НС] RENAME ⟨Имя 1⟩ TO ⟨Имя 2⟩

– меняет старое имя 1 на новое имя 2,

[НС] KILL: F Номер дисковвода ⟨Имя⟩

– уничтожает файл с указанным именем,

[НС] DIR: F [Номер дисковвода]

– выводит каталог дисковвода с указанным номером (если номер не указан, он подразумевается нулевым),

FIELD *#N*, Число байт AS ⟨Текстовая переменная⟩

– присваивает текстовой переменной значение, заданное числом байт файла с номером *N* (FIELD – функция),

MERGE ": F*N*: Имя"

– соединяет программный файл с дисковвода под номером *N* с указанным именем и текущей программы.

Приведенные выше сведения рассчитаны на достаточно опытного пользователя. При работе с файлами нужно исключить возможность случайной записи или стирания файлов. Рекомендуется поупражняться в этом, используя вначале чистый диск; желательно не использовать для записи и диски, хранящие системные программы (интерпретатор Бейсика, различные тексты и сложные прикладные программы). Целесообразно сразу же продублировать их на дополнительных дисках.

6.5. ПОДКЛЮЧЕНИЕ К ПЭВМ ДИСПЛЕЯ И ПРИНТЕРА (ДИРЕКТИВЫ LLIST, LPRINT, COPY)

Подключение дисплея. Подключение к ПЭВМ специализированного дисплея никаких затруднений не вызывает. Надо лишь найти необходимый кабель и нужные разъемы для его подключения на самой ПЭВМ и дисплее.

При использовании в качестве дисплея обычного телевизора прежде всего нужно выяснить, какой сигнал вырабатывает ПЭВМ. Если у нее имеется выход телевизионного *радиосигнала,* то он через коаксиальный кабель подключается к антенному входу телевизора. Надо также знать, по какой системе (PAL, SECAM, NTSC) передается телевизионный радиосигнал с ПЭВМ. Как уже отмечалось, цветная графика возможна, только если ПЭВМ и телевизор относятся к одной из этих систем. В противном случае изображение будет двухцветным, даже если число строк и кадров у систем (PAL и SECAM) одинаково.

Необходимо также, чтобы телевизор работал в том диапазоне частот (каналов), на котором генерирует телевизионный радиосигнал ПЭВМ. Часто это дециметровый диапазон, который имеют не все телевизоры. Наконец, четкость изображения у черно-белых телевизоров обычно значительно выше, чем у цветных. Поэтому, если цвета не обязательны, лучше использовать черно-белый телевизор. Отметим, что отечественные черно-белые телевизоры, работающие по системе SECAM, можно подключать к ПЭВМ, предназначенным для работы по системе PAL (в этом случае частоты строк и кадров идентичны). ПЭВМ, вырабатывающие сигналы по системе NTSC, нельзя подключать к телевизорам системы PAL и SECAM.

Многие телевизоры имеют вход *видеосигнала.* Четкость изображения при подключении ПЭВМ к этому входу обычно выше, чем при подключении ко входу радиосигнала. Отпадает и необходимость в настройке телевизора на нужный канал и в подстройке его в процессе эксплуатации. Однако и в этом случае нужно проверить, согласуется ли выход видеосигнала ПЭВМ со входом его у телевизора. Передача видеосигнала может осуществляться различными способами: передачей полного видеосигнала, цветоразностных сигналов и раздельно сигналов всех цветов. Поэтому не спешите подключать телевизор, прежде чем не выясните, каким именно способом передается видеосигнал и как согласуются разъемы и параметры видеосигнала ПЭВМ и телевизора.

Директивы управления дисплеем были рассмотрены выше. Поэтому ограничимся приведенной выше информацией и рассмотрим некоторые особенности подключения и эксплуатации принтера.

Подключение принтера. Принтер подключается к ПЭВМ непосредственно или через специальный интерфейсный блок, часто с помощью многожильного плоского кабеля. Высококвалифицированные специалисты могут подключить практически любой принтер к ПЭВМ, разработав для этого специальный интерфейс. Однако для большинства пользователей не рекомендуется подключать к ПЭВМ принтеры, специально не предназначенные для работы с данной ПЭВМ. Это чревато серьезными повреждениями как ПЭВМ, так и принтера.

Принтер – сложное электронно-механическое устройство, требует систематического ухода и аккуратной эксплуатации. Уход за принтером сводится к смазке движущихся частей, очистке их от остатков красящей ленты и обрывков бумаги, периодической замене красящей ленты и т. д. Правила эксплуатации принтера описаны в инструкции по его применению. Им нужно следовать.

Немало хлопот пользователю доставляет высыхание красящих лент (особенно, если в принтере применены нестандартные ленты шириной, отличной от 13 и 16 мм). Ленты покрываются специальной эмульсионной краской. Полезно приобрести ее, что позволит обновлять ленты и использовать их многократно с высоким качеством печати (если ленты не на тканевой основе). Для нанесения краски на ленту можно использовать красящие ролики.

Для выдачи на принтер листинга программ и данных служат директивы LLIST и LPRINT. Формы их задания совершенно аналогичны директивам LIST и PRINT (нередко вместо директив LLIST и LPRINT используются директивы LIST и PRINT с особым модификатором, например #1 – включение принтера, #0 – его выключение). В некоторых ПЭВМ есть очень полезная директива COPY – с ее помощью копируются полностью все изображения с рабочей части экрана дисплея (листинги программ, результаты вычислений, графики и т. д.).

Следует позаботиться о программном обеспечении ПЭВМ для работы с принтером. Имеются различные специальные программы для этого: установки различных шрифтов печати, изменения числа знаков в строке, печати графиков с увеличенной в 2 или 3 раза длиной (вдоль рулона бумаги) и соответственно с улучшенным разрешением, проверки технического состояния принтера и т. д.

6.6. МОДЕРНИЗАЦИЯ И РЕМОНТ ПЭВМ

Персональные ЭВМ весьма сложное техническое устройство. Однако при соблюдении определенных мер предосторожности вполне возможна модернизация ПЭВМ силами пользователя-непрофессионала. Разумеется, если есть возможность пригласить для консультации специалиста, лучше сделать это с его помощью.

Что же может сделать обычный пользователь? Не так уж мало. Прежде всего, конструкция многих ПЭВМ предусматривает наличие нескольких свободных разъемов

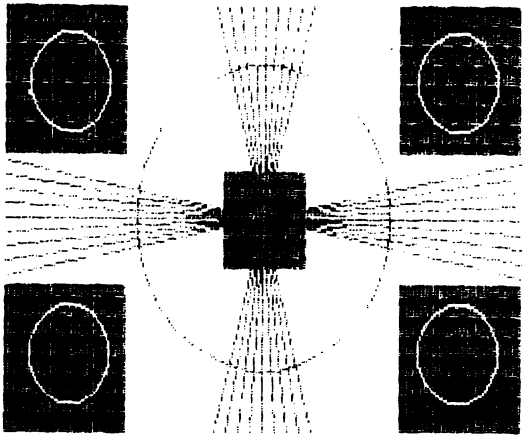


Рис. 6.1

для подключения различных дополнительных устройств: накопителей на магнитных дисках, адаптеров для дисплеев, синтезаторов речи, внешних ПЗУ с библиотеками программ пользователя, дополнительных ОЗУ для расширения памяти ПЭВМ и т.д. Обычно все эти устройства монтируются в системный блок ПЭВМ. Надо проверить, действительно ли эти устройства могут применяться в данной ПЭВМ (указания на это содержатся в эксплуатационной документации на ПЭВМ, но ее не всегда в состоянии осилить рядовой пользователь).

Что касается ремонта ПЭВМ, то он сводится лишь к устранению простейших и достаточно явных неисправностей: замене перегоревших предохранителей, очистке спиртом контактов клавишного пульта, замене отдельных плат (если они имеются в комплекте запасных частей). Многие ПЭВМ имеют специальные диагностические тесты, позволяющие найти неисправную печатную плату или даже отдельную микросхему. Однако эти тесты предназначены для квалифицированных специалистов, осуществляющих централизованное сервисное обслуживание и ремонт ПЭВМ. С помощью этих тестов они могут в течение 10–30 мин найти и заменить на новый неисправный узел, который затем уже ремонтируется в специализированных мастерских.

Даже если пользователь не намерен самостоятельно ремонтировать ПЭВМ, ему полезно приобрести указанные выше диагностические и тестовые программы (в их названии обычно стоит слово TEST). Периодическое испытание ПЭВМ и периферийного оборудования по таким программам – лучшая гарантия безошибочной и правильной работы ПЭВМ длительное время.

Программа TEST позволяет проверить работу клавиатуры ПЭВМ, исправность ОЗУ и ПЗУ, работу с магнитофоном и принтером, оценить общую работоспособность ПЭВМ и др.

Программа TECTTV служит для проверки технических характеристик телевизора или дисплея разрешающей способности, искажений растра (рис. 6.1, например, выявляет растяжение изображения принтером по оси Y), правильность цветопередачи, стабильность синхронизации и т.д. Распечатка графическим принтером изображения с экрана телевизора была получена при работе этой программы (применялась команда COPY). Как видно, эта программа позволяет проверить качество работы не только телевизора, но и принтера (хотя для последнего есть свои тестовые программы, осуществляющие более полную проверку).

Таким образом, ПЭВМ с подобной тестовой программой заменяет генераторы телевизионных сигналов и может успешно использоваться для ремонта и отладки телевизионных приемников.

Программа TESTTAPE (или иная подобная данной) служит для проверки работы магнитофона. Нередко такая программа позволяет получать на экране дисплея осциллограмму сигналов, считываемых с магнитофона, и их специальную спектрограмму. По спектру сигналов можно судить о колебаниях скорости движения магнитной ленты, наличии шумов и фона, степени нелинейных искажений сквозного тракта.

Пользователь, имеющий навыки работы с электронной аппаратурой (пусть даже небольшие) может самостоятельно устранить многие неисправности в ПЭВМ. Однако от этого следует воздержаться в период гарантийного срока, можно лишиться права на гарантийное обслуживание.

Одна из самых распространенных поломок ПЭВМ – отказ в работе отдельных клавиш или их групп. Чаще всего эта неисправность связана с нарушением контактов в самих клавишах или в их соединениях с основной платой (или блоком) ПЭВМ. Эти виды неисправностей вполне поддаются исправлению. Так, нарушение контактов может быть связано с загрязнением контактных частей (их нужно промыть спиртом), отгибанием пружинных контактов (их нужно правильно подогнуть), отпайкой соединений в разъемах (нужно проверить их тестером и при нарушении пайки перепаять разъем), обрывом проводов в соединительном кабеле (также надо "прозвонить" кабель и заменить или отремонтировать его в случае отказа).

У многих современных бытовых ПЭВМ клавиатура подключается через плоские разъемы к основной плате с помощью гибкого плоского кабеля, ленточные проводники которого нанесены на тонкую пластиковую ленту. Этот кабель делается по длине с некоторым запасом и внутри ПЭВМ изгибается в форме буквы S. Обрыв проводников плоского кабеля из-за растрескивания со временем пластиковой основы вблизи разъемов – одна из наиболее часто встречающихся неисправностей, что может быть связано со слишком резким изгибом кабеля у разъемов. Если плоский кабель имеет достаточный припуск по длине, то достаточно отключить кабель от разъема, отрезать дефектную часть кабеля и вновь вставить укороченный кабель в разъем. Затем нужно очень внимательно просмотреть, как укладывается кабель и исключить резкие перегибы его. Если оставшийся отрезок кабеля слишком короткий, следует заменить его целиком (другим плоским кабелем или в крайнем случае жгутом из эластичных обычных проводников).

Часто необходимо состыковать ПЭВМ с имеющимся в распоряжении пользователя кассетным магнитофоном. Обычно это не специальный магнитофон для записи программ и данных, тогда может возникнуть необходимость в изготовлении или перепайке соединительного кабеля. Нужно выход ПЭВМ соединить с микрофонным входом магнитофона, а выход последнего с входом ПЭВМ, найти выход ПЭВМ можно по сигналу, который появляется на нем, если дать команду SAVE "Имя". В качестве индикатора можно использовать включенный на микрофонный вход магнитофон в режиме записи.

Достаточно опытные пользователи могут провести существенную модернизацию ПЭВМ, например изготовить платы с дополнительными микросхемами ОЗУ, ввести звуковой контроль (если его нет) дополнительное энергопитание ОЗУ при отключении ПЭВМ от сети, изготовить синтезаторы речи, подключить обычный телевизор по видеосигналу к ПЭВМ без выхода последнего (это заметно улучшает качество изображения), изготовить специализированный магнитофон, например, с быстрым поиском нужных программ и т.д. Все это требует высокой радиолюбительской или инженерной квалификации и большой аккуратности. Надо тщательно взвесить риск от такого вмешательства и пенять на себя, если дорогая ПЭВМ будет повреждена.

Глава 7

НЕКОТОРЫЕ ПРИМЕНЕНИЯ ПЕРСОНАЛЬНЫХ ЭВМ

7.1. ПЕРСОНАЛЬНЫЕ ЭВМ В РОЛИ ЗАПИСНОЙ КНИЖКИ

Хранение и поиск всевозможных справок, адресов, телефонных номеров, названий предприятий и организаций и т.д. очень удобно поручить ПЭВМ. Тем более, что она обеспечивает поиск подобной информации весьма оперативно и точно.

Существуют специальные программы, например DATEBASE (база данных), предназначенные для накопления, хранения, поиска и выдачи разнообразной информации. Если, однако, их нет, пользователь может самостоятельно составить более простые программы такого рода, например программу, выполняющую роль записной книжки.

Ниже описаны простейшие приемы составления программы "Записная книжка", которая может постоянно пополняться и изменяться и неопытным пользователем. При этом совершенно не нужно следить за алфавитным порядком фамилий или других признаков, по которым выполняется поиск, их можно вносить в программу в произвольном порядке. Фрагмент подобной программы:

```
  10 REM ЗАПИСНАЯ КНИЖКА
  20 INPUT "ФАМИЛИЯ ?";F$
  30 IF F$="АЛЕКСАНДРОВ" OR F$="
А" THEN PRINT "АЛЕКСАНДРОВ И.М Т
ЕЛ.123-45-67"
  40 IF F$="ПЕТРОВ" OR F$="П" TH
EN PRINT "ПЕТРОВ В.М. ТЕЛ.234-45
-67"
  50 IF F$="ПРАВДИН" OR F$="П" T
HEN PRINT "ПРАВДИН И.С. ТЕЛ.456-
67-89:МОСКВА ГОРЬКОГО 34 КВ.23"
  60 IF F$="ПИРОГОВ" OR F$="П" T
HEN PRINT "ПИРОГОВ Э.С, ТЕЛ.789-
56-34:ЛЕНИНГРАД НЕВСКИЙ ПР.12 КВ
.45"
  70 IF F$="ЛАВРОВ" OR F$="Л" TH
EN PRINT "ЛАВРОВ Н.Т. ТЕЛ.562-45
-12"
9999 GO TO 20
```

По запросу ФАМИЛИЯ ? ответ пользователя присваивается символьной переменной F$ (строка 20). Разумеется, можно задать не только фамилию, но и название предприятия или адрес и т.д. Ответ пользователя может содержать только первую букву фамилии или всю фамилию. В первом случае программа выдает справки о всех лицах, фамилии которых начинаются с указанной буквы, во втором – справку только об одном лице.

Вся предыдущая часть программы содержит однотипные команды вида

НС IF F$ = "⟨Фамилия⟩" OR "⟨Первая буква фамилии⟩"
THEN PRINT "⟨Справка о данном лице⟩"

Ответ ПЭВМ на указанную пользователем букву П:

```
ФАМИЛИЯ ? П
ПЕТРОВ В.М. ТЕЛ.234-45-67
ПРАВДИН И.С. ТЕЛ.456-67-89:МОСКВ
А ГОРЬКОГО 34 КВ.23
```

```
ПИРОГОВ З.С. ТЕЛ.789-56-34 ЛЕНИН
ГРАД НЕВСКИЙ ПР.12 КВ.45

ФАМИЛИЯ ? ПРАВДИН
ПРАВДИН И.С. ТЕЛ.456-67-89: МОСКВ
А ГОРЬКОГО 34 КВ.23
```

выведены справки о всех лицах (разумеется, вымышленных), фамилия которых начинаются с буквы П. Зададим затем (в служебной строке) запрос на фамилию ПРАВДИН. Если ввести его (нажав клавишу ввода), будут выведены данные только на лицо с этой фамилией.

Этот простой пример показывает, что составление подобных программ вполне под силу начинающему пользователю ПЭВМ. Введя вместо оператора PRINT оператор GOSUB, можно каждую справку оформить подпрограммой, содержащей более обширную и разнообразную информацию, вплоть до табличной и графической. Таким образом, построение информационных систем тоже не вызывает трудностей.

Подобным способом, но используя оператор LOAD "Имя" LINE⟨Номер строки⟩ или подобный ему, можно построить и информационные системы с поиском информации на магнитном диске или ленте. В этом случае удобнее использовать диск, поскольку время поиска достаточно мало. Некоторые рекомендации по составлению программ такого рода, обеспечивающих ускоренный поиск информации, даны в § 4.5.

7.2. ЗАДАНИЕ ДОПОЛНИТЕЛЬНЫХ МАТЕМАТИЧЕСКИХ ФУНКЦИЙ

Набор математических функций у большинства версий языка Бейсик ограничен лишь основными элементарными функциями действительной переменной. Он должен быть существенно дополнен, если ПЭВМ применяется для научно-технических и инженерных расчетов.

Дополнительные элементарные функции. Могут вычисляться по отдельным программам [1]. Удобнее использовать для этого библиотеки подпрограмм. В табл. 7.1 представлены наиболее распространенные дополнительные функции и блок подпрограмм для их вычисления.

Таблица 7.1

Базовые функции универсальной подпрограммы (блок 1)

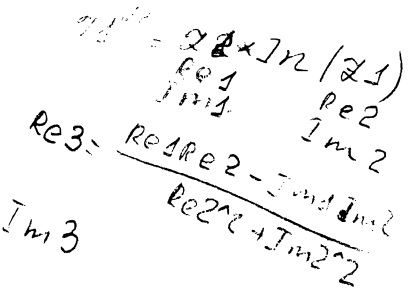
Функции	Расчетное выражение	Аргумент	Обращение в программе
Гиперболические	$\operatorname{sh} x = (e^x - e^{-x})/2$	X	FN Q( )
	$\operatorname{ch} x = (e^x + e^{-x})/2$	X	FN W( )
	$\operatorname{th} x = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$	X	FN E( )
Обратные гиперболические	$\operatorname{arsh} x = \ln(x + \sqrt{x^2 + 1})$	X	FN A( )
	$\operatorname{arch} x = \ln(x + \sqrt{x^2 - 1})$	X	FN S( )
	$\operatorname{arth} x = \ln\sqrt{(1 + x)/(1 - x)}$	X	FN D( )

Окончание табл. 7.1

Функции	Расчетное выражение	Аргумент	Обращение в программе
Модуль и фаза комплексного числа $Z = X + iY$	$M = \sqrt{X^2 + Y^2}$ $\varphi = F = \operatorname{sgn}(Y)\arccos(X/M)$	X X	FN M( ) FN F( )
$f = Z^2$	$\operatorname{Re} f = X^2 - Y^2$ $\operatorname{Im} f = 2XY$	X, Y X, Y	FN H( ) FN J( )
$f = 1/Z$	$\operatorname{Re} f = X/(X^2 + Y^2)$ $\operatorname{Im} f = -Y/(X^2 + Y^2)$	X, Y X, Y	FN N( ) FN V( )
$f = 1/Z^2$	$\operatorname{Re} f = (X^2 - Y^2)/(X^2 + Y^2)$ $\operatorname{Im} f = 2XY/(X^2 + Y^2)$	X, Y X, Y	FN G( ) FN Z( )
$f = \sqrt{Z}$	$\operatorname{Re} f = \sqrt{(X + \sqrt{X^2 + Y^2})/2}$ $\operatorname{Im} = \sqrt{(-X + \sqrt{X^2 + Y^2})/2}$	X, Y X, Y	FN C( ) FN P( )
$f = \ln Z$	$\operatorname{Re} = (\ln(X^2 + Y^2))/2$ $\operatorname{Im} = \operatorname{arctg}(Y/X)$	X, Y X, Y	FN B( ) FN O( )
sh Y	$\operatorname{sh} Y = (e^Y - e^{-Y})/2$	Y	FN K( )
ch Y	$\operatorname{ch} Y = (e^Y + e^{-Y})/2$	Y	FN L( )
Параметр B	$B = (\sqrt{(X+1)^2 + Y^2} - \sqrt{(X-1)^2 + Y^2})/2$	X, Y	FN I( )
Параметр A	$A = (\sqrt{(x+1)^2 + Y^2} + \sqrt{(X-1)^2 + Y^2})/2$	X, Y	FN U( )
$f = \operatorname{arctg} Z$	$\operatorname{Re} f = \frac{1}{2}\operatorname{arctg}\left(\frac{2X}{1 - X^2 - Y^2}\right)$	X, Y	FN T( )
	$\operatorname{Im} f = \frac{1}{4}\ln\left(\frac{X^2 + (Y+1)^2}{X^2 + (Y-1)^2}\right)$	X, Y	FN R( )

```
8000 REM ПОДПРОГРАММЫ ВЫЧИСЛЕНИЯ
8001 REM ЭЛЕМЕНТАРНЫХ ФУНКЦИЙ
8002 REM ДЕЙСТВИТЕЛЬНОГО И КОМПЛ
     ЕКСНОГО АРГУМЕНТА
8003 REM **********************
8004 REM ГИПЕРБОЛИЧЕСКИЕ ФУНКЦИИ
8005 DEF FN Q()=(EXP X-EXP -X)/2
8006 DEF FN W()=(EXP X+EXP -X)/2
8007 DEF FN E()=(EXP X-EXP -X)/(
EXP X+EXP -X)
8008 REM ОБРАТНЫЕ ГИПЕРБОЛИЧЕСКИ
Е ФУНКЦИИ
8009 DEF FN A()=LN (X+SQR (X*X+1
))
```

```
8010 DEF FN S()=LN (X+SQR (X*X-1
))
8011 DEF FN D()=LN ((1+X)/(1-X))
/2
8012 REM МОДУЛЬ И ФАЗА КОМПЛЕКСН
ОГО ЧИСЛА
8013 DEF FN M()=SQR (X*X+Y*Y)
8014 DEF FN F()=SGN (Y)*ACS (X/S
QR (X*X+Y*Y))
8015 REM RE(Z↑2),IM(Z↑2)
8016 DEF FN H()=X*X-Y*Y
8017 DEF FN J()=2*X*Y
8018 REM RE(1/Z),IM(1/Z)
8019 DEF FN N()=X/(X*X+Y*Y)
8020 DEF FN V()=-Y/(X*X+Y*Y)
8021 REM RE(1/Z↑2),IM(1/Z↑2)
8022 DEF FN G()=(X*X-Y*Y)/(X*X+Y
*Y)↑2
8023 DEF FN Z()=2*X*Y/(X*X+Y*Y)↑
2
8024 REM RESQR (Z),IMSQR (Z)
8025 DEF FN C()=SQR ((X+SQR (X*X
+Y*Y))/2)
8026 DEF FN P()=SQR ((-X+SQR (X*
X+Y*Y))/2)
8027 REM RELN (Z),IMLN (Z)
8028 DEF FN B()=LN (X*X+Y*Y)/2
8029 DEF FN Q()=ATN (Y/X)
8030 REM HSN(Y),HCN(Y)
8031 DEF FN K()=(EXP Y-EXP -Y)/2
8032 DEF FN L()=(EXP Y+EXP -Y)/2
8033 REM B,A ДЛЯ ASN(Z),ACS(Z)
8034 DEF FN I()=(SQR ((X+1)*(X+1
)+Y*Y)-SQR ((X-1)*(X-1)+Y*Y))/2
8035 DEF FN U()=(SQR ((X+1)*(X+1
)+Y*Y)+SQR ((X-1)*(X-1)+Y*Y))/2
8036 REM REATN (Z),IMATN (Z)
8037 DEF FN T()=ATN (2*X/(1-X*X-
Y*Y))/2
8038 DEF FN R()=LN ((X*X+(Y+1)*(
Y+1))/(X*X+(Y-1)*(Y-1)))/4
```



Если, например, нужно вычислить гиперболический синус $\mathrm{sh}\, x$ при $x = 1$, достаточно ввести

```
LET X = 1 : PRINT FN Q ( )
```

Получим результат $\mathrm{sh}\,(1) = 1{,}1752012$.

Функции комплексной переменной. В электрорадиотехнических расчетах широко применяются функции комплексной переменной $Z = x + \mathrm{i}y$, где x – действительная и y – мнимая части. Операторы DEF FN и FN приходится использовать парами, так как функция FN обычно имеет один выход, а результат (комплексное число) представляется парой чисел. Например, функция возведения Z в квадрат определяется как $Z^2 = (x^2 - y^2) + \mathrm{i}(2xy)$. Часть таких функций включена в табл. 7.1 и блок программ. Так, для вычисления $(3 + \mathrm{i}2)^2$ достаточно задать:

```
LET X = 3 : LET Y = 2 : PRINT FN N( ) : PRINT FN V( )
```

Получим результат $Z = 5 + \mathrm{i}2$ (результат заменяет аргумент Z). Другие функции можно вычислить, воспользовавшись известными формулами вычисления функций комплексной переменной [1, 3, 32, 33].

Для создания библиотек функций удобны расширенные версии Бейсика с аппаратом задания и вызова поименованных процедур. Это позволяет отождествлять имена процедур с общепринятыми обозначениями функций и пользоваться ими наряду со встроенными функциями. Некоторые версии Бейсика допускают такие обозначения в командах DEF FN.

Специальные функции. Эти функции широко используются в науке и технике. Так, в теории спектров широко применяется интегральный синус $\mathrm{Si}(x)$, в статистике – гамма-функция $\Gamma(x)$ и т.д. Они не имеют представления через элементарные функции и обычно вычисляются численными методами. В отличие от таблиц специальных функций [32, 33], вычисление их на ПЭВМ возможно при любых (а не только табличных) значениях аргумента без применения интерполяции. Ограничимся кратким описанием основных методов вычисления таких функций.

Рекуррентные выражения. Рекуррентными называют выражения, очередной член которых вычисляется через предыдущие. Классический пример – вычисление факториала $N! = (N-1)!\,N$ (см. гл. 3). Аналогично вычисляется гамма-функция: $\Gamma(x+1) = \Gamma(x)x$. Этот прием позволяет свести аргумент x в область значений, где удобно использовать другие методы, например аппроксимацию функций, разложение в ряд и т.д. Для $\Gamma(x)$ пример этого дан ниже.

Максимальное значение $x = N$ при вычислении $N!$ ограничено у большинства ПЭВМ значениями $N = 34$: при $N > 34$ наступает переполнение разрядной сетки (обычно значения $35! = 1{,}033 \cdot 10^{40}$ уже недопустимы). В некоторых случаях желательно вычислять факториалы значительно бóльших чисел. Если задавать результат как $x! = x_1 \cdot 10^{y_1}$, то факториал больших чисел можно вычислять по такому алгоритму:

1. Задаем $y_0 = 0$.

2. Организуем цикл $y_i = y_{i-1} + \lg i = y_{i-1} + \ln i/\ln 10$ с управляющей переменной i, меняющейся от начального значения 1 до конечного x с шагом 1.

3. При выходе из цикла вычисляем $y_1 = \mathrm{int}\, y$, $x_1 = 10^{(y-y_1)}$ и получаем $x!$ в форме $x_1 \cdot 10^{y_1}$.

Этот алгоритм реализует следующая подпрограмма:

```
8120 REM ПОДПРОГРАММА ВЫЧИСЛЕНИЕ
 X!
8121 IF X>=34 THEN GO TO 8124
8122 LET Y=1: FOR I=1 TO X
8123 LET Y=Y*I: NEXT I: LET Y1=0
: RETURN
8124 LET Y=0: FOR I=1 TO X
8125 LET Y=Y+LN I/2.30258509: NE
XT I
8126 LET Y1=INT Y: LET Y=10↑(Y-Y
1): RETURN
8127 PRINT "X=";: INPUT "ВВЕДИТЕ
 X=";X: PRINT X;: GO SUB 8121
8128 IF X<34 THEN PRINT "  X!=";
Y: PAUSE 0: GO TO 8127
8129 PRINT "  X!=";Y;"*E";Y1: PA
USE 0: GO TO 8127
```

Для $x = 100$ получим $100! = 9{,}3326282\,\mathrm{E}\,157$ (этим результатом можно пользоваться как конечным, но не как обычным числом).

Разложение в ряд – наиболее удобный способ вычисления специальных функций. Так, при $x < 10$ интегральный синус вычисляется сходящимся рядом

$$\mathrm{Si}(x) = \sum_{i=0}^{\infty} \frac{(-1)^i x^{2i+1}}{(2i+1)(2i+1)!}.$$

Подпрограмма, вычисляющая Si x по разложению в ряд с заданной погрешностью $\epsilon = E$:

```
8188 REM ВЫЧИСЛЕНИЕ ФУНКЦИИ SI(X
)
8189 LET X1=X: LET SI=X: LET X3=
-(X*X)/2: LET I=0
8190 LET I=I+1: LET X2=(I+I+1):
LET X2=X2*X2
8191 LET X1=((I+I-1)*X3*X1)/I/X2
8192 LET SI=SI+X1: IF ABS X1)=1E
-9 THEN GO TO 8190
8193 RETURN
8194 PRINT "X=";: INPUT "X=";X:
PRINT X;: GO SUB 8189
8195 PRINT "  SI(X)=";SI: PAUSE
0: GO TO 8194
```

Так, при $\epsilon = 1 \cdot 10^{-6}$ и $x = 1$ получим Si(1) = 0,846083.

Аппроксимация специальных функций производится с помощью полиномов, элементарных функций и др. Пример подпрограммы вычисления $\Gamma(x) = G(x)$ при $0 \leqslant x \leqslant 1$ с помощью аппроксимирующего полинома:

```
8206 REM ВЫЧИСЛЕНИЕ ФУНКЦИИ G(X)
ПО АППРОКСИМАЦИИ
8207 LET X1=ABS X: LET X2=1
8208 IF X1<=1 THEN GO TO 8210
8209 LET X2=X2*X1: LET X1=X1-1:
GO TO 8208
8210 LET Y=((.035868343*X1-.1935
27818)*X1+.482199394)*X1
8211 LET Y=(((Y-.756704078)*X1+.
918206857)*X1-.897056937)*X1
8212 LET Y=((Y+.988205891)*X1-.5
77191652)*X1+1
8213 LET G=Y*X2/X: IF X<0 THEN G
O TO 8215
8214 RETURN
8215 LET G=PI/SIN (PI*X)/X2/Y: R
ETURN
8216 PRINT "X=";: INPUT "ВВЕДИТЕ
 X=";X: PRINT X;: GO SUB 8207
8217 PRINT "  G(X)=";G: PAUSE 0:
 GO TO 8216
```

При $x > 1$ аргумент x с помощью рекурсии приводится в область $0 \leqslant x \leqslant 1$, а при $x < 0$ используется формула $\Gamma(x) = \pi / (\sin \pi|x| \, \Gamma(|x| + 1))$. Так, при обращении к этой подпрограмме получим $\Gamma(-3{,}2) = 0{,}689056$.

Асимптотические выражения дают значения функций, выражаемые через элементарные функции, при $x \to 0$ или $x \to \infty$. Так, $\Gamma(x)$ при $x \to \infty$ определяется формулой Стирлинга

$$\Gamma(x) \approx \sqrt{\frac{2\pi}{x}}\, e^{-x} x^x \left(1 + \frac{1}{12} + \frac{1}{288x^2} - \frac{139}{51\,840x^3} \ldots\right).$$

Прибавив к малым x некоторое целое число n, $\Gamma(x+n)$, можно довольно точно вычислять по формуле Стирлинга, ограничившись 2–3 членами в круглых скобках. Затем $\Gamma(x+n)$ можно свести к $\Gamma(x)$, применяя рекуррентный процесс вниз. На этом и построена следующая программа:

```
8258 REM ВЫЧИСЛЕНИЕ ГАММА-ФУНКЦИ
И G(X)  С ПРИМЕНЕНИЕМ АСИМПТОТИЧЕ
СКОГО РАЗЛОЖЕНИЯ(X<=23.8)
8259 LET X1=X: LET A=1: LET Y=X
8260 FOR I=1 TO 10
8261 LET Y=Y*(X1+I): NEXT I: LET
 X2=X1+11
```

```
8262 LET G=(EXP X2)↑(LN X2-1)*EX
P (1/12/X2-1/360/X2↑3+1/1260/X2↑
5)
8263 LET G=G*SQR (2*PI/X2)/Y: RE
TURN
8264 PRINT "X=";: INPUT "ВВЕДИТЕ
 X=";X: PRINT X;
8265 GO SUB 8259: PRINT "  G(X)=
";G: PAUSE 0: GO TO 8264
```

7.3. МАТЕМАТИЧЕСКИЕ РАСЧЕТЫ (РЕАЛИЗАЦИЯ ОСНОВНЫХ ЧИСЛЕННЫХ МЕТОДОВ

Персональная ЭВМ – прекрасный инструмент для проведения математических расчетов – от табуляции формул до реализации специальных численных методов. Поскольку реализация численных методов на Бейсике достаточно подробно описана в [1], остановимся лишь на некоторых наиболее важных методах, программная реализация их дана в приложении 1.

Численное интегрирование. Выполняется чаще всего методом Симпсона при аппроксимации подынтегральной функции $f(x)$ отрезками парабол. Используется формула

$$I = \int_a^b f(x) \approx \frac{h}{3}[f(a) + 4f(a+h) + 2f(a+2h) + 4f(a+3h) + \ldots$$

$$\ldots + 4f(b-h) + f(b)].$$

Интервал интегрирования делят на $m = 2, 4, 8, \ldots$ сдвоенных отрезков и вычисляют $I_1, I_2, I_3, \ldots$. При каждом делении погрешность уменьшается в 15 раз. Вычисления прерываются, если $|I_{n-1} - I_n| \leqslant 15\varepsilon$, где ε – заданная погрешность.

Решение систем нелинейных уравнений. Решение систем нелинейных уравнений, имеющих в матричной форме вид $\mathbf{F(X)} = 0$, чаще всего выполняется итерационным методом Ньютона–Рафсона (см. § 3.2). Он может быть реализован на ПЭВМ матричными операторами. Программа, реализующая этот метод с помощью обычных версий Бейсика, дана в приложении 1 (программа П1.2). С помощью подобных программ можно решать и одно уравнение вида $f(x) = 0$.

Решение алгебраических уравнений. Алгебраическими обычно называют уравнения вида

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 = 0,$$

где $P(x)$ – полином. Следовательно, решение таких уравнений дает корни полинома. Трудность реализации решения подобных уравнений связана с тем, что корни $P(x) = 0$ в общем случае являются комплексными. Это очевидно уже из рассмотрения решения приведенного квадратного уравнения $x^2 + a_1 x + a_0 = 0$, которое имеет вид

$$x_{1,2} = x \pm iy, \text{ где } x = -a_1/2 \text{ и } y = \sqrt{(a_1/2) - a_0}.$$

Для решения уравнения $P(x) = 0$ при любой степени полинома и действительных коэффициентах полинома можно использовать метод Хичкока, который заключается в следующем. При $n = 1$ имеем единственное ре-

шение $x_1 = -a_0/a_1$. При $n = 2$ решение рассмотрено выше. При $n \geqslant 3$ можно задать два приближения x_1 и x_2 и применить процедуру минимизации $P(x)$. Значения x_1 и x_2 при $P(x) \to 0$ можно считать парой найденных корней. Разделив $P(x)$ на квадратичный множитель $(x - x_1)(x - x_2) = (x^2 + a_1 x + a_0)$, можно понизить на 2 порядок $P(x)$ и вернуться к анализу остатка, и так, пока не будут найдены все корни полинома. Отметим, что нахождение комплексных корней полинома – важнейшая задача при расчете электронных цепей операторным методом.

Решение систем дифференциальных уравнений. Наиболее распространенным методом решения дифференциальных уравнений

$$y' = \frac{dy}{dx} = f(x, y)$$

на ПЭВМ является метод Рунге–Кутта, при котором новое значение y_{i+1} решения $y(x)$ находится по старому y_i с помощью рекуррентного выражения

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

где $k_1 = hf(x_i y_i)$, $k_2 = hf(x_i + h/2, y_i + k_1/2)$, $k_3 = hf(x_i + h/2, y_i + k_2/2)$, $k_4 = hf(x_i + h, y_i + k_3)$; h – шаг изменения аргумента x.

Для старта (начала вычислений) надо задать начальные значения x_0 и y_0 (начальные условия). Эта процедура легко распространяется и на решение систем дифференциальных уравнений (см. программу П1.4). Ошибка метода Рунге–Кутта пропорциональна h^5, при больших h возможна численная неустойчивость решения.

Операции с полиномами. Важное значение полиномов, в частности, для аппроксимации функций уже не раз отмечалось. Поэтому полезно иметь программу для действий с полиномами. Сложение и вычитание двух полиномов $A(x)$ и $B(x)$ сводится просто к сложению и вычитанию их коэффициентов с одинаковыми степенями.

В программе П1.5 обеспечивается вычисление значений полиномов, а также умножение и деление двух полиномов. Для умножения полиномов результирующий полином имеет степень $n + m$. При делении полиномов $A(x)/B(x)$ при $n > m$ получается полином степени m и остаток в виде полинома $R(x)$ степени $m - n$. Операции с полиномами выполняются по правилам, описанным в [33], при представлении полиномов схемой Горнера. С алгоритмом этих операций нетрудно разобраться, рассмотрев соответствующие фрагменты упомянутой программы:

Номера строк	Назначение
10–28	Ввод коэффициентов полинома $A(x)$
30–70	Задание "меню" операций
80–110	Вычисление производной полинома $A(x)$
120–150	Вычисление интеграла полинома $A(x)$
160–180	Ввод коэффициентов полинома $B(x)$
190–220	Вычисление $C(x) = A(x)\ B(x)$

Окончание табл.

Номера строк	Назначение
230–270	Вычисление $C(x) = A(x)/B(x)$ и остатка $R(x)$
280–290	Вычисление значения $A(x)$ при заданном x

Вычисление производных полинома. Дифференцируя полином вида

$$A(x) = \sum_{i=0}^{n} a_i x^i,$$

получим первую производную

$$A'(x) = \frac{dA(x)}{dx} = \sum_{i=1}^{n} a_{i-1}^{i} i x^{i-1}.$$

Она имеет вид полинома степени $n-1$, причем каждый член этого полинома вычисляется как $a_{i-1}i$. Этот процесс можно повторять для вычисления второй, третьей и т. д. производных.

Вычисление неопределенного интеграла полинома. Если $A(x)$ – подынтегральная функция, то

$$\int A(x)\,dx = \int \left(\sum_{i=0}^{n} a_i x^i\right) dx = \sum_{i=0}^{n} \frac{a_i}{i+1} x^{i+1}.$$

Таким образом, неопределенный интеграл от многочлена есть полином степени $n+1$, причем его коэффициенты равны $a_i/(i+1)$.

При вычислении производной и интеграла мы получаем результат в близком к аналитическому виде – как новый полином с рассчитанными коэффициентами. Вычислив значения полученного полинома при заданном x, нетрудно получить конкретное значение производной $A'(x)$. Аналогично, вычислив полином для $x = b$ и $x = a$ на основании формулы

$$\int_a^b A(x)\,dx = B(x)\,|_{x=b} - B(x)\,|_{x=a},$$

получаем значение определенного интеграла (здесь $B(x)$ обозначен полином – неопределенный интеграл).

Включение аналитических операций в функциональные возможности ПЭВМ связано со значительным усложнением как их аппаратной части, так и (это главное) процедуры общения с ПЭВМ. Поэтому современные ПЭВМ, как правило, не могут выполнять аналитические операции по встроенным микропрограммам. Вероятно, в будущем профессиональные ПЭВМ (ориентированные на математиков, физиков и ученых другого профиля), выполняющие аналитические операции, получат широкое распространение.

Наряду с обычными полиномами, находят применение и другие специальные виды полиномов: ортогональные, полиномы наилучшего приближения Чебышева, сплайны и др. С ними можно подробно познакомиться по книгам [1, 29 – 33]. Некоторые виды таких полиномов применительно к решению задач машинной графики рассматриваются в следующем параграфе.

7.4. ПОСТРОЕНИЕ ГРАФИКОВ ПРОИЗВОЛЬНЫХ ФУНКЦИЙ И ИХ АППРОКСИМАЦИЯ

Одна из привлекательных возможностей ПЭВМ – построение графиков всевозможных функций. Такие графики могут строиться с координатными осями и без них, с поясняющими надписями и различными цветовыми и световыми эффектами (например, с разным цветом линий, миганием их). Существует несколько способов построения графиков функций, указанных ниже.

Построение графиков по точкам. В этом случае задается изменение аргумента x в заданных пределах и вычисляется ряд значений x_i (обычно в цикле). По этим значениям находят $y_i = f(x_i)$, где $y = f(x)$ – требуемая для построения графика зависимость. Допустимы только $x_i > 0$ и $y_i > 0$, поскольку экран дисплея отображает только первый квадрант декартовой системы координат. Однако, добавив к x_i и y_i координаты точки центра x_0 и y_0 новой системы координат, всегда можно сдвинуть график в нужную часть экрана. Этот метод иллюстрируется рис. 7.1.

График содержит $256/2 = 128$ точек и строится довольно медленно – несколько секунд. Несмотря на большое число точек, он имеет вид прерывистой кривой. Значение y_i задается выражением $60 \sin(x/25) + 70$, где 60 – амплитуда синусоиды (выражена в числе пикселей); $x/25 = \omega x$ (ω – круговая частота; $70 = y_0$ – смещение кривой вверх на 70 пикселей).

Построение графиков с помощью векторов. Этот способ соответствует линейной аппроксимации функции в промежутках между двумя ее отсчетами. Он легко реализуется на ПЭВМ, имеющих операторы PLOT – TO, LINE и DRAW для построения отрезков прямых-векторов. Обычно операторы PLOT – TO строят график, соединяя две точки с координатами (x_{i-1}, y_{i-1}) и (x_i, y_i) отрезком прямой. Поэтому в программе построения

```
10 REM ПОСТРОЕНИЕ СИНУСОИДЫ С
ПОМОЩЬЮ ОПЕРАТОРА PLOT
20 FOR X=0 TO 255 STEP 2
30 PLOT X,60*SIN (X/25)+70
40 NEXT X
```

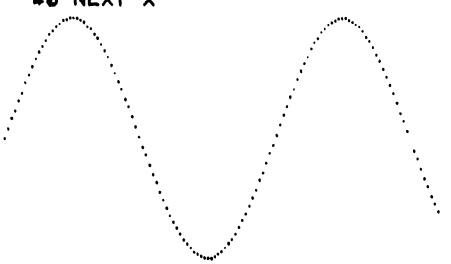


Рис. 7.1

```
 5 LET z=75: PLOT 0,75
10 FOR x=5 TO 250 STEP 5
20 LET y=60*SIN (x/25)+75
30 DRAW 5,y-z
40 LET z=y: NEXT x
50 FOR x=0 TO 255 STEP 25
60 PLOT x,75: NEXT x
70 FOR y=0 TO 150 STEP 25
80 PLOT 50,y: NEXT y
```

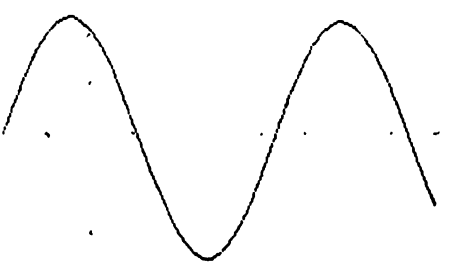


Рис. 7.2

```
  5 REM ПОСТРОЕНИЕ КООРДИНАТНОЙ
СЕТКИ
  10 FOR X=20 TO 220 STEP 20
  20 PLOT X,20: DRAW 0,100: NEXT
X
  30 FOR Y=20 TO 120 STEP 10
  40 PLOT 20,Y: DRAW 200,0: NEXT
Y
  50 PRINT AT 20,1;"0";AT 14,1;"
4"
  60 PRINT AT 9,1;"8";AT 6,1;"Y"
  70 PRINT AT 20,7;"2";AT 20,12;
"4"
  80 PRINT AT 20,17;"6";AT 20,22
8"
  90 PRINT AT 20,28;"X"
```

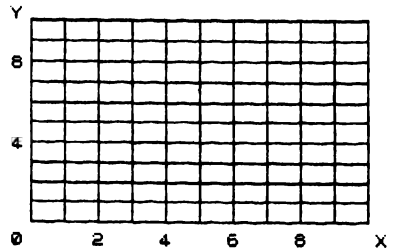


Рис. 7.3

графика вначале строится точка x_0, y_0 оператором PLOT x_0, y_0, а затем в цикле используется оператор PLOT x_{i-1}, y_{i-1} TO x_i, y_i (либо LINE x_i, y_i).

Перемещение очередной точки на заданные приращения $\Delta x_i = x_i - x_{i-1}$ и $\Delta y_i = y_i - y_{i-1}$ обеспечивает оператор DRAW. Он особенно удобен, если $\Delta x_i = \text{const}$, т.е. значения x_0, x_1, $x_2, \ldots, x_i$ идут с одинаковым шагом Δx_i. Построение синусоиды с помощью оператора DRAW иллюстрирует рис. 7.2. Хотя график построен всего по 25 узловым точкам, он выглядит лучше, чем график на рис. 7.1. Это наглядное свидетельство в пользу даже простейшей аппроксимации $y = f(x)$ в промежутках между узлами.

Построение координатных осей. Наглядность графика улучшается, если он построен на фоне координатных осей. Их можно наметить в виде точек (рис. 7.2) либо сформировать с помощью специальной программы (см. рис. 7.3). В последнем случае фактически строится сетка с координатными осями. Для этого удобно использовать циклы, формирующие вертикальные (строки 10, 20) и горизонтальные (строки 30, 40) линии сетки. Остальная часть программы формирует надписи.

Задание графика массивом данных. В процессе построения графика функции ее приходится многократно вычислять. Нередко, особенно в обучающих программах, программах-справочниках и т.д., необходимо в нужный момент быстро построить график какой-то сложной функции. В этом случае предварительно вычисленные значения функции $y_i(x_i)$ целесообразно запомнить в виде массива данных. Тогда построение графика выполняется с помощью следующей программы:

```
HC1  FOR I=1 TO N
HC2  PLOT X(I),Y(I) : NEXT I
```

Такой подход (см. пример построения эллипса § 5.5) заметно увеличивает скорость построения графиков, но ценой затрат памяти ПЭВМ на хранение массивов x_i и y_i (при равномерном расположении узлов хранить все x_i не нужно, так как $x_i = x_0 + i\Delta x$, т.е. достаточно хранить x_0 и Δx).

Полиномиальная аппроксимация при построении графиков. Как отмечалось (см. § 3.1), всегда можно построить полином $(n-1)$-й степени, проходящий через n узлов функции $y = f(x)$. Такой полином обычно вычисляется намного быстрее, чем функция $f(x)$, которую он аппроксимирует. Нельзя ли воспользоваться этим и заменить график $f(x)$ графиком поли-

нома? В целом ответ положителен, однако только для простых функций. Уже при степени полинома выше 5–6 он нередко ведет себя непредсказуемо в промежутках между узлами. В результате получается характерная волнистость, не свойственная самой функции $f(x)$.

Кроме того, степень полинома однозначно связана с числом отсчетов функции (узлов). По мере усложнения функции степень полинома растет, растут затраты времени на вычисление каждой точки графика и погрешности аппроксимации. Например, N-образная вольт-амперная характеристика туннельного диода (см. рис. 3.2) требует для аппроксимации применения полинома не менее 5-й степени. Поэтому в программах построения графиков функций обычные полиномы применяются редко.

Многоинтервальная полиномиальная аппроксимация. Недостатки полиномов можно заметно ослабить, уменьшив степень полинома. Но тогда график полинома нужно проводить не через все узловые точки, а лишь через часть их. А если проводить аппроксимацию рядом полиномов одинаковой степени, но с разными для разных участков $y = f(x)$ коэффициентами? Оказывается, в преимуществе такого метода мы уже могли убедиться. Действительно, построение графика с помощью отрезков прямых-векторов есть не что иное, как применение полинома первой степени

$$y(x)_{i+1} = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i} x = y_i + a_{i+1} x$$

на каждом интервале $[x_i, x_{i+1}]$ изменения аргумента x.

Нетрудно понять, что применение полинома степени 2 или 3 значительно улучшит совпадение полинома с нелинейной функцией на небольшом отрезке изменения ее аргумента. Особенно большой эффект дает специальная многоинтервальная аппроксимация, рассмотренная ниже. Здесь и далее отсчет i будет идти не с 0, а с 1.

Сплайн-аппроксимация. Сплайны – полиномы степени m, коэффициенты которых при многоинтервальной аппроксимации выбираются такими, чтобы обеспечить непрерывность до $m-1$ производной в точках стыка. Заметим, что непрерывность нулевой производной означает совпадение точек стыка с узловыми точками. Таким образом, аппроксимация сплайном первого порядка, в сущности, есть кусочно-линейная аппроксимация.

Чаще всего применяется сплайн третьего порядка. Если он дает непрерывность первых производных, то это означает равенство крутизны сплайн-функции по обе стороны от каждого узла. Сплайн с непрерывной первой и второй производной можно мысленно (и весьма наглядно!) представить в виде гибкой линейки, закрепленной в узловых точках. Другими словами, сплайн – линия, наиболее естественным образом проходящая через узловые точки (любопытно, что сплайн с нулевой второй производной в конечных точках так и называется "естественным").

Сплайн-функция третьего порядка в каждом интервале изменения x $[x_i, x_{i+1}]$ задается в виде [33]

$$f(x) = \frac{1}{6h_i}\,[m_i(x_{i+1} - x)^3 + m_{i+1}(x - x_i)^3] + \frac{1}{h_i}\,[(y_i -$$

$$- \frac{m_i h_i^2}{6})(x_{i+1} - x) + (y_{i+1} - \frac{m_{i+1} h_i^2}{6})(x - x_i)], \qquad (7.1)$$

где $h_i = (x_{i+1} - x_i)$; $m_i = f''(x_i)$, $i = 1, 2, 3, \ldots, n$. Такой вид удобен для вычисления коэффициентов m_i.

Задавшись $f_i(x_i) = y_i$ из этого выражения, можно получить систему линейных уравнений

$$h_i m_i + 2(h_i + h_{i+1}) m_{i+1} + h_{i+1} m_{i+2} = 6(\frac{y_{i+2} - y_{i+1}}{h_{i+1}} - \frac{y_{i+1} - y_i}{h_i}),$$

позволяющих найти $m_2 \ldots m_{n-1}$. Если задать оставшиеся ненайденными $m_1 = m_n = 0$, получаем нормальную, или естественную, сплайн-функцию. Если $m_n = m_1$, $m_{n+1} = m_2$, – периодическую и т. д.

На рис. 7.4 дана программа сплайн-аппроксимации и интерполяции с построением графика аппроксимирующей функции.

Сначала (строки 20–150) вычисляются коэффициенты $m_1 - m_n$ сплайн-функции по заданным n узловым точкам (x_i, y_i) табличной, аналитической или графической зависимости $y(x)$. Затем (строки 160–210) выполняется сплайн-интерполяция – по заданному x из (7.1) находится $f(x)$. Для последующего использования вычисления сплайн-функции оформлены в виде подпрограммы (строки 180–210). Если не предполагается использовать программу для построения графиков, ее можно оборвать на строке 210. Таким образом, строки 20–210 образуют программу сплайн-аппроксимации и интерполяции, имеющую самостоятельное значение.

Фрагмент программы от строки 220 до 290 служит для построения графика сплайн-функции туннельного диода, имеющего параметры вольт-амперной характеристики (ВАХ), заданные 7 узлами (число узлов можно менять):

i	1	2	3	4	5	6	7
x = U, В	0	0,2	0,4	0,6	0.8	1	1,2
y = I, мА	0	50	20	3	4	14	55

График функции $I(U)$ = y (x) строится по 45 точкам (строки 220–260) с применением для этого обращения к подпрограмме вычисления сплайн-функции (строки 180–210, обратите внимание: в данном случае подпрограмма расположена в середине общей программы). Строки 270 и 280 формируют координатные оси, строка 290 – поясняющую надпись.

При пуске (командой RUN) программа запрашивает число узлов n, затем значения $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$. После ввода их (см. выше) происходит вычисление коэффициентов m_i = MI. Для данного примера программа выдает $m_0 - m_7$:

```
  10 PRINT """""СПЛАЙН-АППРОКСИ
МАЦИЯ ГЛОБАЛЬНАЯ": PAUSE 50: CLS

  20 PRINT "ЧИСЛО УЗЛОВ  n=";: I
NPUT "ВВЕДИТЕ ЧИСЛО УЗЛОВ n=";n:
 PRINT n
  30 DIM x(n): DIM y(n): DIM l(
n): DIM m(n): DIM r(n): DIM s(n)
  40 FOR i=1 TO n: PRINT "x";i;"
=";: INPUT "ВВЕДИТЕ x(i)=";x(i):
 PRINT x(i)
  50 PRINT "y";i;"=";: INPUT "ВВ
ЕДИТЕ y(i)=";y(i): PRINT y(i): N
EXT i
  60 LET d=x(2)-x(1): LET e=(y(2
)-y(1))/d
  70 FOR k=2 TO n-1: LET h=d: LE
T d=x(k+1)-x(k)
  80 LET f=e: LET e=(y(k+1)-y(k)
)/d: LET l(k)=d/(d+h)
  90 LET r(k)=1-l(k): LET s(k)=6
*(e-f)/(h+d): NEXT k
 100 FOR k=2 TO n-1: LET p=1/(r(
k)*l(k-1)+2): LET l(k)=-l(k)*p
 110 LET s(k)=(s(k)-r(k)*s(k-1))
*p: NEXT k
 120 LET m(n)=0: LET l(n-1)=s(n-
1): LET m(n-1)=l(n-1)
 130 FOR k=n-2 TO 1 STEP -1
 140 LET l(k)=l(k)*l(k+1)+s(k):
LET m(k)=l(k): NEXT k
 150 FOR k=1 TO n: PRINT "M";k;"
=";m(k): NEXT k
 160 INPUT "G ИЛИ Y?";f$: IF f$=
"g" OR f$="G" THEN GO TO 220
 170 PRINT "x=";: INPUT "ВВЕДИТЕ
 x=";x: PRINT x;: LET i=0: GO SU
B 180: PRINT "  y(x)=";y: GO TO
 160
 180 LET i=i+1: IF x>x(i) THEN G
O TO 180
 190 LET j=i-1: LET d=x(i)-x(j):
 LET h=x-x(j): LET r=x(i)-x
 200 LET p=d*d/6: LET y=(m(j)*r↑
3+m(i)*h↑3)/6/d
 210 LET y=y+((y(j)-m(j)*p)*r+(y
(i)-m(i)*p)*h)/d: RETURN
 220 CLS : LET z=20: PLOT 20,20
 230 FOR a=25 TO 250 STEP 5
 240 LET i=0: LET x=(a-20)/200
 250 GO SUB 180: LET y1=2*y+20:
DRAW 5,y1-z
 260 LET z=y1: NEXT a
 270 PLOT 20,20: DRAW 230,0: PLO
T 220,20: DRAW 0,5: PRINT AT 20,
27;"1 B": PLOT 120,20: DRAW 0,5:
 PRINT AT 20,14;"0.5"
 280 PLOT 20,0: DRAW 0,140: PLOT
 20,120: DRAW 5,0: PRINT AT 6,0;
"50": PRINT AT 3,0;"mA"
 290 PRINT AT 19,0;"0": PRINT AT
 2,7;"ВАХ ТУННЕЛЬНОГО ДИОДА"
```

Рис. 7.4

M1 = 0
M2 = 3304, 4231
M3 = 1217, 6923
M4 = 383,65385
M5 = –52,3007692
M6 = 1175,5769
M7 = 0

После этого делается запрос: G или y. Если нужно выполнить интерполяцию, следует указать y (вычисление y(x) по заданному x). Тогда появляется указание

ВВЕДИТЕ x =

Например, если x = U = 0,3В, получим y = I = 40,216827 мА. Если нужно вывести график, следует указать G. Результат выполнения для этого случая показан на рис. 7.4.

Сплайн-аппроксимация является мощным математическим приемом, с помощью которого можно решать множество расчетных и графических задач дифференцирования и интегрирования функций, решения нелинейных уравнений, расчета переходных процессов в электронных цепях с нелинейными вольт-амперными характеристиками и т. д. Задав цикличекое выполнение программы сплайн-аппроксимации, можно выполнить аппроксимации семейства кривых [1]. Сплайн-аппроксимация обеспечивает высокую точность при малом числе узлов и при широком общем интервале изменения аргумента $[x_1, x_n]$.

7.5. ПОСТРОЕНИЕ РИСУНКОВ НА ПЛОСКОСТИ И В АКСОНОМЕТРИИ

Построение рисунков на плоскости. К широко распространенным простейшим рисункам на плоскости относятся гистограммы. Обычно гистограмма изображается в виде столбиков, высота которых характеризует средний уровень величин, для которых дана гистограмма. Этими величинами могут быть производительность предприятия, рост или возраст людей, количество годных изделий и т. д.

Данные для гистограммы обычно получают в виде массива $Y(I)$, где Y – высота столбца; I – его номер. Если число столбцов N, а интервал изменения аргумента X задан равным 200 пикселям, то приращение $\Delta X = X_1$, определяющее ширину каждого столбца, должно быть равно $\Delta X_1 = 200/N$. Программа для быстрого построения гистограмм по заданному числу столбцов N и массиву $Y(I)$ и результат ее выполнения приведены на рис. 7.5.

В программе для построения столбцов использованы операторы задания векторов DRAW *x, y*. Построение гистограммы из 10 столбцов производится буквально в доли секунды.

Восприятие изображения на плоскости сильно зависит от способов его построения и размеров изображения. Для иллюстрации этого рассмотрим изображение обычной спирали (рис. 7.6), формируемое в параметрической форме

$$x = r \sin \omega + 120,$$
$$y = r \cos \omega + 80,$$

где переменная ω изменяется от 0 до 30 с шагом 0,1 и $r = 2\omega$. В этом случае число точек значительно и спираль имеет естественный вид (хотя и четко заметна точечная структура изображения).

Построение расходящейся спирали (рис. 7.7) достигается возрастанием параметра r по закону $r = 1 \cdot e^{\omega/25}$. Число точек изображения уменьшено до 100. Это наглядный пример оптической иллюзии, связанной с конечным числом точек графика: создается впечатление, что рисунок содержит 6 расходящихся спиралей, а вовсе не одну!

Еще один пример (рис. 7.8) иллюстрирует построение сходящейся спирали с параметром $r = 50(1 - e^{-\omega/20})$. Число точек здесь велико – 500. С первого взгляда спираль напоминает скорее стилизованную сферу.

Шестеренка, приведенная на рис. 7.9, получается в результате построения "окружности" с переменным быстро изменяющимся радиусом $r = A = 50 + 8 \sin(12N)$, где N – управляющая переменная цикла. "Окружность" строится по параметрическим уравнениям $x = 130 + A \cos N$ и $y = 60 + A \sin N$. В конце программы (строка 50) строится обычная окружность – внутреннее отверстие шестеренки.

Окружность как место расположения особых точек геометрических фигур широко используется для их построения. Например, многоугольник может строиться следующим образом (рис. 7.10): на окружности, в которую вписывается N-угольник, выбирается N равноотстоящих точек, затем они соединяются по кругу друг с другом отрезками прямой. Программа

```
  10 REM ГИСТОГРАММА
  20 PRINT "ВВЕДИТЕ N=";: INPUT
"N=";N: PRINT N
  30 DIM Y(N): FOR I=1 TO N
  40 PRINT "ЗАДАЙТЕ Y";I;"=";: I
NPUT "Y=";Y(I): PRINT Y(I): NEXT
 I
  50 CLS : PLOT 20,15: DRAW 210,
0
  60 PLOT 20,15: DRAW 0,130
  70 LET X1=200/N: LET X=0: LET
Y=0
  80 FOR I=1 TO N: PLOT X+20,15
  90 DRAW 0,Y(I): LET X=X+X1: IF
 Y(Y(I) THEN LET Y=Y(I)
 100 DRAW X1,0: DRAW 0,-Y(I): NE
XT I
 110 PRINT AT 0,5;"ДЕЛЬТА X=";X1
 120 PRINT AT 0,20;"Y МАКС=";Y
```

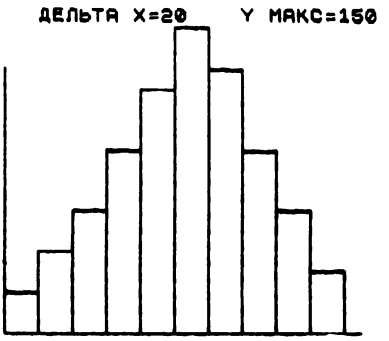


Рис. 7.5

```
  10 REM СПИРАЛЬ ОБЫЧНАЯ
  20 FOR W=0 TO 30 STEP .1
  30 LET X=SIN W: LET Y=COS W: L
ET R=2*W
  40 PLOT 120+R*X,80+R*Y: NEXT W
```

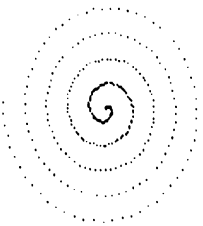


Рис. 7.6

```
  10 REM СПИРАЛЬ РАСХОДЯЩАЯСЯ
  20 FOR W=0 TO 100 STEP 1
  30 LET X=SIN W: LET Y=COS W: L
ET R=1*EXP (W/25)
  40 PLOT 120+R*X,80+R*Y: NEXT W
```

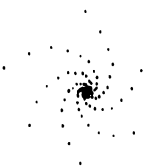


Рис. 7.7

```
  10 REM СПИРАЛЬ СХОДЯЩАЯСЯ
  20 FOR W=0 TO 100 STEP .2
  30 LET X=SIN W: LET Y=COS W: L
ET R=50*(1-EXP (-W/20))
  40 PLOT 120+R*X,80+R*Y: NEXT W
```

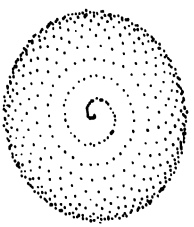


Рис. 7.8

```
10 REM ШЕСТЕРЕНКА
20 FOR N=0 TO 2*PI STEP PI/180
25 LET A=50+8*SIN (N*12)
30 PLOT 130+A*COS N,80+A*SIN N
40 NEXT N
50 CIRCLE 130,80,15
```

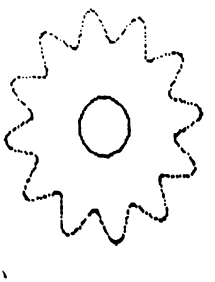


Рис. 7.9

обеспечивает построение N-угольника по заданному пользователем значению N.

Еще более оригинальные фигуры получаются при построении точек на окружности, которые затем соединяются друг с другом (каждая точка соединяется со всеми другими, но сама окружность не строится) [34]. Подобная программа приведена на рис. 7.11, а внизу результат ее выполнения при N= 5 и N= 15. Второй рисунок уже весьма необычен, складывается впечатление о его сложности и особой рельефности, хотя от первого рисунка он отличается только числом исходных точек.

Приведенные выше программы иллюстрируют построение геометрических фигур на плоскости. Разумеется, пользователь всегда может по точкам или по отрезкам прямых и дуг построить вначале рисунок вручную, а затем запрограммировать его для построения на экране дисплея ПЭВМ. Такие операции элементарны, и нет необходимости на них останавливаться особо. Отметим лишь, что в этом случае часто полезно преобразование масштаба изображения, если геометрические размеры исходного рисунка отличаются от размеров рисунка на экране дисплея ПЭВМ.

Пусть объект (стрелка на рис. 7.12) расположен в прямоугольном окне, заданном в координатной системе пользователя координатами левого нижнего угла (u_1, v_1) и верхнего правого угла (u_2, v_2). Его необходимо перенести на экране дисплея, расположив в кадре с проблемно-ориентированными координатами (x_1, y_1) и (x_2, y_2) соответствующих углов. Чтобы обеспечить такой перенос в общем случае достаточно координаты u и v любой точки исходного объекта преобразователь в координаты x и y по формулам [35]:

$$x = x_1 + (u - u_1)K_x,$$
$$y = y_1 + (v - v_1)K_y,$$

где $K_x = (x_2 - x_1)/(u_2 - u_1)$; $K_y = (y_2 - y_1)/(v_2 - v_1)$.

Координаты x, y обычно называют растровыми, поскольку они задаются целыми числами в растровых единицах экрана дисплея. Однако практически нет необходимости выделять целые части x и y, рассчитанных по приведенным формулам, поскольку обычно это делается автоматически при использовании графических операторов, например PLOT.

Аксонометрические изображения объектов. Строятся в трехмерной системе координат (рис. 7.13). В действительности изображение на экране дисплея, как и на листе бумаги, всегда двухмерное. Следовательно, речь идет лишь об иллюзии создания объемных изображений.

Каждый из нас может быстро нарисовать простой объект в аксонометрии. Нередко перенос такого объекта на экран дисплея ПЭВМ – наиболее простой метод воспроизведения его. Задача упрощается, если воспользоваться изображением объекта, нанесенным на разлинованный в клетку лист бумаги, в котором выделено поле с указанием растровых единиц. Тогда остается лишь засечь растровые координаты характерных точек объекта и составить программу их построения и соединения друг с другом. Аналогично строятся дуги, окружности и эллипсы. На рис. 7.14 приведены про-

```
  10 REM  МНОГОУГОЛЬНИК
  20 INPUT "ВВЕДИТЕ  N=";N
  30 LET X=130: LET Y=102: LET D
=2*PI/N: PLOT X,Y
  40 FOR I=0 TO 2*PI+.1 STEP D
  50 LET X1=130+50*SIN (I)
  60 LET Y1=52+50*COS (I)
  70 DRAW X1-X,Y1-Y
  80 LET X=X1: LET Y=Y1: NEXT I
```

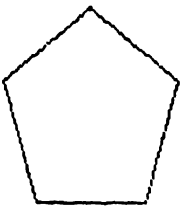
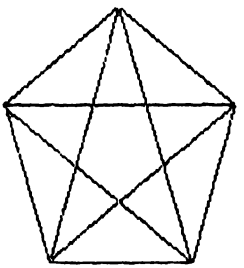


Рис. 7.10

```
  10 REM ПОСТРОЕНИЕ N ТОЧЕК НА
ОКРУЖНОСТИ СОЕДИНЕННЫХ ДРУГ С
ДРУГОМ
  20 INPUT "ВВЕДИТЕ ЧИСЛО ТОЧЕК
N=";N: DIM X(N): DIM Y(N)
  30 LET D=2*PI/N: LET T=0
  40 FOR I=1 TO N: LET T=T+D
  50 LET X(I)=130+70*SIN (T)
  60 LET Y(I)=80+70*COS (T): NEX
T I
  70 LET M=N-1: FOR I=1 TO M
  80 LET I1=I+1: FOR J=I1 TO N
  90 PLOT X(I),Y(I): DRAW X(J)-X
(I),Y(J)-Y(I)
 100 NEXT J: NEXT I
```



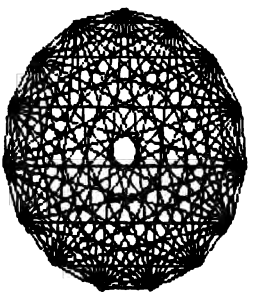


Рис. 7.11

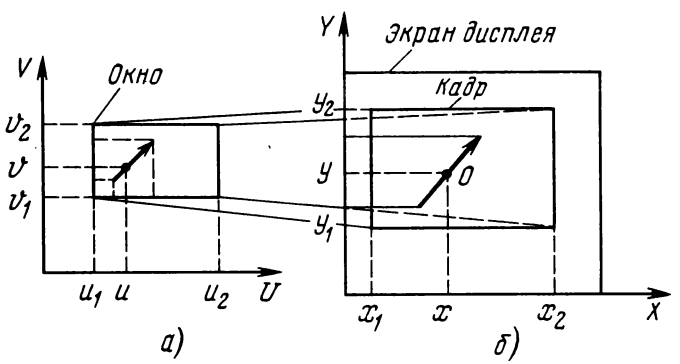


Рис. 7.12

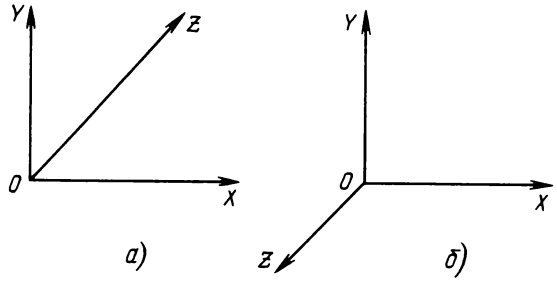


Рис. 7.13

```
  10 REM ПОСТРОЕНИЕ ДОМИКА
  15 REM ФАСАД ДОМА
  20 PLOT 30,55: DRAW 0,45: DRAW
 80,0: DRAW 0,-45: DRAW -80,0
  30 PLOT 30,100: DRAW 40,30: DR
AW 40,-30
  35 REM ЧЕРДАЧНОЕ ОКНО
  40 CIRCLE 70,110,6
  45 REM БОКОВАЯ СТЕНА
  50 PLOT 110,100: DRAW 90,40: D
RAW 0,-45: DRAW -90,-40
  55 REM КРЫША
  60 PLOT 70,130: DRAW 90,35: DR
AW 40,-25
  65 REM ОКНО ФАСАДА
  70 PLOT 85,70: DRAW 0,20: DRAW
 20,0: DRAW 0,-20
  80 DRAW -20,0: PLOT 85,85: DRA
W 20,0
  90 PLOT 95,90: DRAW 0,-20
  95 REM КРЫЛЬЦО
 100 PLOT 40,55: DRAW 0,4: DRAW
30,0: DRAW 0,-4
 110 PLOT 40,57: DRAW 30,0
 115 REM ДВЕРЬ
 120 PLOT 45,89: DRAW 0,30: DRAW
 20,0: DRAW 0,-30
 130 PLOT 62,72: DRAW 0,6
 135 REM БОКОВЫЕ ОКНА
 140 PLOT 125,80: DRAW 0,20: DRA
W 15,6: DRAW 0,-20: DRAW -15,-6
 150 PLOT 125,95: DRAW 15,6: PLO
T 132,83: DRAW 0,20
 160 PLOT 175,102: DRAW 0,20: DR
AW 15,6: DRAW 0,-20: DRAW -15,-6
 170 PLOT 175,117: DRAW 15,6: PL
OT 182,105: DRAW 0,20
 175 REM  ПЕЧНАЯ ТРУБА
 180 PLOT 50,115: DRAW 0,10: DRA
W 7,0: DRAW 0,-6
```

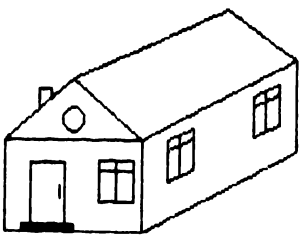


Рис. 7.14

```
  10 REM  ПИРАМИДА
  20 INPUT "ВВЕДИТЕ  N=";N
  30 LET X=130: LET Y=77: LET D=
2*PI/N
  40 FOR I=0 TO 2*PI+.1 STEP D
  50 LET X1=130+50*SIN (I)
  60 LET Y1=52+25*COS (I)
  70 PLOT (X,Y): DRAW (X1-X,Y1-Y)
  80 PLOT (X,Y): DRAW (130-X,160-Y)
  90 LET X=X1: LET Y=Y1: NEXT I
```

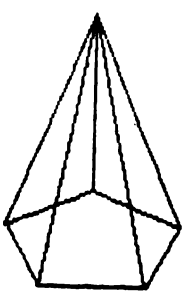


Рис. 7.15

грамма построения изображения домика, составленная по этому рецепту, и сам домик, на построение которого по этой программе ПЭВМ затрачивает доли секунды.

Элементарные геометрические фигуры часто можно строить с помощью специальных программ. Ниже дано несколько таких примеров.

Пирамида может быть построена следующим образом. В растровых координатах строится N точек эллипса, затем эти точки соединяются друг с другом и с точкой – вершиной пирамиды. Этот алгоритм реализован программой на рис. 7.15.

Рис. 7.16 иллюстрирует построение конуса с помощью оператора DRAW x, y, r, который строит ряд дуг (строки 20–40) разной длины. В строке 60 строятся образующие конуса. Такое построение не вполне точное: дуга окружности неэквивалентна дуге эллипса. В отличие от построения пирамиды в данном случае отсутствуют невидимые линии.

Рис. 7.17 иллюстрирует построение стилизованного объемного изображения цилиндра или трубы. Здесь строятся два полуэллипса (левый и правый), на них выбирается ряд точек, соединяемых горизонтальными прямыми. Левый полуэллипс достраивается до эллипса переносом точек из его правой части в левую (учитывается симметрия фигуры).

В общем случае положение любой точки объекта в трехмерном пространстве задается тремя координатами x, y и z. Однако лишь координаты x и y являются действительными. Ось Z является воображаемой осью. Расположенные на этой оси точки выражаются через их координаты на действительных осях X и Y, т. е.

$$z = \sqrt{x^2 + y^2} \text{ или } x = z \cos \varphi,\ y = z \sin \varphi.$$

Здесь φ – угол, который образует ось Z с осью X.

Таким образом, задание координаты z эквивалентно изменению координат x и y. Например, если ось Z образует с осью X угол $\varphi = 45°$, то $\sin \varphi = \cos \varphi = 1/\sqrt{2} \approx 0{,}707$. Следовательно, чтобы построить точку на оси Z с координатой $(0,\ 0,\ z)$ достаточно задать $x_Z = z \cos \varphi \approx 0{,}707z$ и $y_Z = z \cos \varphi \approx 0{,}707z$.

В общем случае координаты точек по осям X, Y и Z являются линейно независимыми. Следовательно, точка с координатами (x, y, z) в системе растровых координат может быть построена как точка с координатами на плоскости

$$X = x_0 + x + 0{,}707\,z,\ Y = y_0 + y + 0{,}707\,z,$$

где x_0 и y_0 – координаты центра декартовой системы.

Этот простой и эффективный прием иллюстрирует программа построения синусоиды, лежащей в плоскости X, Z (рис. 7.18).

Строятся координатные оси X, Y и Z (строки 20–70). Затем организуется цикл управляющей переменной W, изменяющейся от 0 до 4π с шагом $\pi/50$ (строка 80). Последнее обеспечивает построение двух периодов синусоиды из 200 точек. Синусоида строится по уравнениям $x = 40 \sin W$, $z = 6W$. В строке 90 вычисляются значения x и z, в строке 100 оператором PLOT с аргументами X и Y (при $x_0 = 80$ и $y_0 = 5$) выполня-

```
10 REM ПОСТРОЕНИЕ КОНУСА
20 FOR I=0 TO 20
30 LET X=130-I: LET Y=120-I*5
40 PLOT X,Y: DRAW 2*I,0,1
50 NEXT I
60 DRAW -20,100: DRAW -20,-100
```

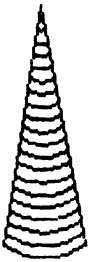


Рис. 7.16

```
10 REM ПОСТРОЕНИЕ ЦИЛИНДРА
20 FOR W=0 TO PI STEP .2
30 LET X=20*SIN (W): LET Y=35*
COS (W)
40 PLOT X+50,Y+42: DRAW 150,0
50 NEXT W
60 FOR W=0 TO 2*PI STEP .05
70 LET X=20*SIN (W): LET Y=35*
COS (W)
80 PLOT X+50,Y+42: IF W<PI THE
N PLOT X+200,Y+42
90 NEXT W
```

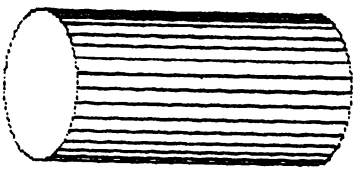


Рис. 7.17

```
10 REM ПОСТРОЕНИЕ СИНУСОИДЫ В
ПЛОСКОСТИ Z,X
20 PLOT 80,5: DRAW 80,0
30 PLOT 80,5: DRAW 0,70
40 PLOT 80,5: DRAW 57,57
50 PRINT AT 20,20;"X"
60 PRINT AT 12,9;"Y"
70 PRINT AT 13,18;"Z"
80 FOR W=0 TO 4*PI STEP PI/50
90 LET Z=6*W: LET X=40*SIN (W)
100 PLOT 80+X+.707*Z,5+.707*Z:
NEXT W
```

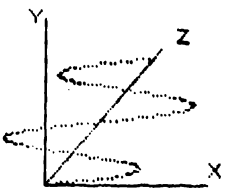


Рис. 7.18

```
10 REM ПОСТРОЕНИЕ ЭЛЛИПСА С ГЛ
АВНОЙ ОСЬЮ ПАРАЛЛЕЛЬНОЙ ОСИ Z
20 PLOT 80,5: DRAW 80,0
30 PLOT 80,5: DRAW 0,70
40 PLOT 80,5: DRAW 57,57
50 PRINT AT 20,20;"X"
60 PRINT AT 12,9;"Y"
70 PRINT AT 13,18;"Z"
80 FOR W=0 TO 2*PI STEP PI/50
90 LET Z=40*SIN (W): LET X=20*
COS (W): LET Y=30
100 PLOT 80+X+.707*Z,5+Y+.707*Z
NEXT W Y
```

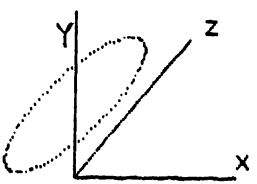


Рис. 7.19

```
10 REM ПОСТРОЕНИЕ ПАРАБОЛОИДА
20 FOR Z=-3 TO 2 STEP .1
30 LET A=.707*Z: LET Z2=Z*Z
40 FOR X=-3 TO 3 STEP .1
50 LET Y=.5*(Z2+X*X)
60 PLOT 130+10*(X+A),10+10*(Y+
A): NEXT X: NEXT Z
```

Рис. 7.20

```
10 REM ФИГУРА ИЗ ОКРУЖНОСТЕЙ
20 LET N=10: LET R=30
30 FOR L=0 TO 2*R STEP 2*R/N
40 LET L1=L*.70711
50 CIRCLE 128,80+L,R
60 CIRCLE 128+L1,80-L1,R
70 CIRCLE 128-L1,80-L1,R
80 NEXT L
```

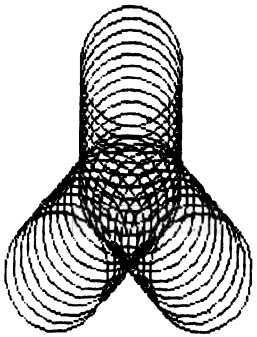


Рис. 7.21

ется собственно построение синусоиды по точкам. Как видно из результата выполнения этой программы (фигура на рис. 7.19 внизу), достигается иллюзия расположения синусоиды в плоскости X, Z.

Для построения эллипса, нанизанного на ось Y (рис. 7.19), используются уравнения

$$x = 20 \cos \omega, z = 40 \sin \omega, y = 30, 0 \leqslant \omega < 2\pi.$$

В действительности эллипс строится по координатам x и y, к которым добавлены члены $0{,}707z$. Большая ось эллипса параллельна оси Z.

С помощью интегральных операционных усилителей и микросхем-квадраторов легко создаются различные функциональные устройства, у которых выходной сигнал U_y является нелинейной функцией двух входных сигналов U_x и U_z. Такие зависимости удобно отображать в виде объемных аксонометрических фигур.

В качестве примера рассмотрим представление зависимости

$$U_y = k(U_x^2 + U_z^2),$$

где k – коэффициент пропорциональности. Для большей общности обозначим сигналы U_y, U_x и U_z как Y, X и Z. Тогда для построения фигуры (в данном случае параболоида) надо использовать уравнение ($k = 0{,}5$)

$$Y = k(X^2 + Z^2).$$

Для значений Z (от -3 до 2 с шагом 0,1) будем задавать значения X (от -3 до 3 с шагом 0,1), затем вычисляем Y. Построение обеспечивается заданием координат x и y, к которым добавлены x_0, y_0 и $0{,}707z$. Соответствующая программа приведена на рис. 7.20. Несимметричное задание z (от -3 до 2) обеспечивает более естественный вид параболлоида.

Возможно и расположение осей X и Y под углом 90° друг к другу и 45° по отношению к вертикальной оси Z. На рис. 7.21 показано построение фигуры из ряда окружностей с обложки книги [35]. Фигура получается последовательным смещением окружности по осям X, Y и Z на расстояние, равное ее диаметру.

7.6. ПОСТРОЕНИЕ ДВИЖУЩИХСЯ ОБЪЕКТОВ

Современные расширенные версии Бейсика имеют специальные графические команды для построения движущихся изображений, их объединения в динамические спрайты и реализации принципов мультипликации (см. § 5.9). Однако в большинстве версий Бейсика этих средств нет. Поэтому рассмотрим некоторые дополнительные возможности создания движущихся изображений.

Рис. 7.22 иллюстрирует процесс движения точки, в ходе которого она описывает фигуры Лиссажу. Пожалуй, это один из немногих случаев, когда медленное движение точки полезно: проследив его, можно лучше понять смысл построений. Фигуры Лиссажу обычно наблюдают с помощью электронного осциллографа, на входы X и Y которого подают два синусоидальных сигнала с кратностью K и фазовым сдвигом Q. Программа на рис. 7.22 моделирует этот процесс (построенная фигура соответствует $K = 3$ и $Q = 0$).

Для перемещения более сложных объектов используется принцип мультипликации; объект строится, наблюдается, стирается, перемещается, вновь строится и т.д. Чтобы создать иллюзию плавного перемещения, надо быстро строить объект. Поэтому в обычных версиях Бейсика строящиеся движущиеся объекты должны быть простыми. Нежелательны их фрагменты в виде дуг, эллипсов и окружностей: из-за многократного вычисления тригонометрических функций они строятся медленно. Такие объекты вместо плавного движения "возникают" прямо на глазах и медленно прыгают. Иногда это создает интересные эффекты, но в целом нежелательно.

Получить эффект перемещения можно для объектов, состоящих из 5–20 прямых линий. Программа "Снаряды", иллюстрирующая принцип мультипликации (рис. 7.23), наглядно показывает преимущества оператора DRAW для построения объектов из отрезков прямых. Действительно, перемещение снаряда по горизонтальной оси требует лишь изменения координат в операторе PLOT (строка 30), задающем базовую точку объекта, от которой идет построение всей фигуры. Во всех последующих операторах DRAW аргументами являются константы. В данном случае с помощью цикла с заголовком (строка 20) строятся пять снарядов, последовательно и быстро (почти мгновенно) появляющиеся на экране.

Программа на рис. 7.24 иллюстрирует уже перемещение снаряда как одного объекта. Весь путь базовой точки объекта (и самого объекта) разбит на 44 этапа. Это достигается заданием цикла (строка 20), в ходе которого координата x базовой точки меняется от 0 до 220 с шагом $\Delta x = 5$.

В программе на рис. 7.24, в отличие от программы на рис. 7.23, построение объекта (снаряда) оформлено в виде программы (строки 50–80). Принцип мультипликации реализован операторами, поочередно повторяемыми 45 раз (строки 30 и 40). Вначале (строка 30) оператор OVER 0 и обращение к подпрограмме GO SUB 50 строят объект с заданными координатами базовой точки. В конце строки 30 задана пауза (с увеличением ее

```
  10 REM ФИГУРЫ ЛИССАЖУ
  20 INPUT "ЗАДАЙТЕ КРАТНОСТЬ ЧА
СТОТ K=";K
  30 INPUT "ЗАДАЙТЕ ФАЗУ (РАД.)
W=";W
  40 FOR I=0 TO 100 STEP .5
  50 LET W=2*PI*I/100
  60 LET X=130+40*SIN (W)
  70 LET Y=45+40*COS (K*W)
  80 PLOT X,Y: NEXT I
```

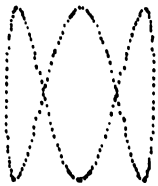


Рис. 7.22

```
10 REM СНАРЯДЫ
20 FOR X=0 TO 200 STEP 50
30 PLOT X,50
40 DRAW 20,0: DRAW 10,6
50 DRAW -10,6: DRAW -20,0
60 DRAW 0,-12: NEXT X
```

Рис. 7.23

```
  10 REM ПЕРЕМЕЩЕНИЕ СНАРЯДА
  20 FOR X=0 TO 220 STEP 5
  30 OVER 0: GO SUB 50: PAUSE 1
  40 OVER 1: GO SUB 50: NEXT X:
GO TO 20
  50 PLOT X,50: DRAW 20,0
  60 DRAW 10,6: DRAW -10,6
  70 DRAW -20,0: DRAW 0,-11
  80 RETURN
```

Рис. 7.24

уменьшается скорость перемещения объекта). Затем (строка 40) с помощью оператора OVER 1 с обращением к подпрограмме тот же объект строится в цвете, совпадающем с цветом общего фона экрана. Другими словами, объект закрашивается. На этом один цикл завершается и начинается новый цикл – повторное выполнение строки 30 (но с новыми координатами), а затем 40.

Работа программы в момент фотосъемки с экрана дисплея ПЭВМ была искусственно приостановлена. В результате мы видим один кадр – неподвижный снаряд в центре экрана. Если пустить программу обычным образом (командой RUN), то на экране будет наблюдаться перемещающийся слева направо небольшими скачками снаряд.

Эта программа иллюстрирует простейший вид движения – вдоль координатной оси X. Очевидно, нетрудно заставить перемещаться снаряд и вдоль оси Y, а воспользовавшись приемами задания координат по оси Z (см. § 7.6), обеспечить перемещение объекта и вдоль оси Z. Таким образом, если известна аналитически траектория перемещения объекта в пространстве, можно заставить перемещаться его аналогичным образом и на экране дисплея. Однако во всех этих случаях вид объекта и его размеры остаются неизменными, нельзя заставить объект поворачиваться, при "удалении" он не будет уменьшаться и т.д. Детальное обсуждение принципов задания произвольного перемещения сложных объектов в пространстве не входит в задачи этой книги.

Достаточно простое изложение основ этих принципов дано в [34], а также [36 – 38].

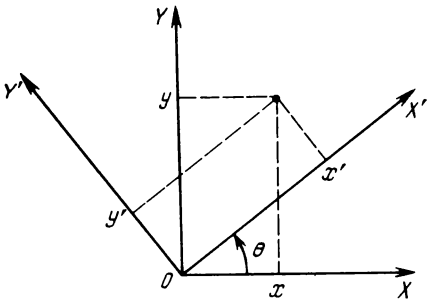


Рис. 7.25

Обычно двухмерное пространство, в котором находится объект, считается неизменяемым. Движения объекта рассматриваются как противоположные по знаку перемещения координатных осей (это позволяет использовать общий математический аппарат преобразования координат), любое перемещение объекта – как совокупность трех преобразований: переноса начала координат, изменения масштаба и поворота осей координат. Рассмотрим это для объектов на плоскости.

Будем рассматривать изменения координат некоторой произвольной точки объекта, происходящие в ходе преобразований систем координат. В действительности объект описывается совокупностью точек, каждая из которых меняет свое положение при переходе от старой системы координат к новой. Таким образом, нужно вычислять координаты всех точек объекта или хотя бы узловых, если объект образован отрезками прямых и дуг, проходящих через узловые точки.

Перенос начала координат. Оси старой и новой систем координат взаимно параллельны и одинаково направлены. Однако новое начало координат в старой системе имеет координаты (T_x, T_y). Следовательно, в новой системе координат начало старой есть точка $(-T_x, -T_y)$. Любая точка (x, y) в прежней системе координат становится точкой $(x - T_x, y - T_y)$ в новой системе координат.

Изменение масштаба. В этом случае оси X и Y старой и новой систем координат совпадают, но масштабы по осям отличаются в S_x и S_y раз. Так, точка (x, y) прежней системы координат будет соответствовать точке (xS_x, yS_y) в новой системе координат.

Поворот осей координат. В этом случае новая и прежняя системы координат имеют одинаковые масштабы по осям, но оси новой системы повернуты на угол θ (рис. 7.25). Из законов аналитической геометрии на плоскости известно, что в этом случае новые координаты (x', y') точки выражаются через прежние (x, y) следующим образом:

$$x' = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$
$$y' = -x \cdot \sin(\theta) + y \cdot \cos(\theta)$$

Матричное преобразование координат точки. Перемещение сложных объектов приходится рассматривать как преобразование каждой точки. Для этого удобен единый матричный математический аппарат, когда точка в двухмерном пространстве представляется вектор-столбцом $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.

Перенос начала координат рассматривается как умножение матрицы переноса на этот вектор:

$$\begin{bmatrix} 1 & 0 & -T_x \\ 0 & 1 & -T_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x - T_x \\ y - T_y \\ 1 \end{bmatrix} .$$

Изменение масштаба задается умножением на вектор матрицы изменения масштаба

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} ,$$

а поворот осей координат – умножением на вектор матрицы поворота осей

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} .$$

Комбинируя операции с этими матрицами, можно получить любое заданное преобразование. Эти преобразования применяются лишь при реализации графики на быстродействующих ЭВМ. Особые трудности представляет поворот осей, поскольку он требует вычисления двух тригонометрических функций. В связи с этим у наиболее совершенных (и дорогих) ПЭВМ описанные преобразования выполняются микропрограммно, при исполнении специальных операторов переноса осей, изменения масштаба и поворота осей. Для большинства ПЭВМ выполнение этих преобразований обычным путем (т.е. с помощью программ на Бейсике или другом языке) ведет к неприемлемо большим временам преобразования и медленному перемещению объекта. По указанным причинам произвольное перемещение объектов, как правило, не задается, а искусственно заменяется простыми видами перемещений. Например, полет снаряда по параболической траектории задается при горизонтальном расположении оси его вращения, бегущий мимо наблюдателя человек изображается всегда строго сбоку (не учитывается его поворот относительно наблюдателя).

Шуточная программа "Посадка летающей тарелки" (рис. 7.26) иллюстрирует технику посадки "летающей тарелки" посреди стилизованного городского ландшафта. "Тарелка" появляется над шоссе и постепенно приближается к нам, увеличиваясь в размерах. Изображение самой тарелки приплюснутого пятиугольника, перечеркнутого горизонтальной чертой, формируется подпрограммой (строки 200–240), причем размеры "тарелки" задаются управляющей переменной основного цикла I (заголовок в строке 50). Тарелка на экране движется вниз и чуть влево, т.е. происходит ее смещение по обеим осям координат с одновременным изменением масштаба изображения. На рисунке показана "тарелка" в момент остановки. "Окна" формируются с помощью графем (как и стилизованные "небоскребы"), причем они появляются в последний момент. Подобный прием

```
  10 REM ПОСАДКА ЛЕТАЮЩЕЙ ТАРЕЛК
И
  15 FOR Y=0 TO 8: PRINT AT Y,9;
"▟▙▟▙      ▟▙▟▙": NEXT Y
  20 PLOT 0,120: DRAW 255,0
  30 PLOT 120,120: DRAW -80,-90
  40 PLOT 130,120: DRAW 80,-90
  50 FOR I=1 TO 6 STEP .2
  60 OVER 1: GO SUB 200
  70 OVER 1: GO SUB 200: NEXT I
  80 OVER 0: GO SUB 200
  90 PRINT AT 21,3;"С БЛАГОПОЛУЧ
НЫМ ПРИБЫТИЕМ"
 100 PRINT AT 17,10;"▟▙▟▙▟▙"
 150 PAUSE 0: CLS : GO TO 10
 200 LET Y=10*I: LET A=5*I: LET
B=2*I: LET C=4*I
 210 PLOT 125-C,120-Y: DRAW A,-B
 220 DRAW -B,-C: DRAW -Y+C,0
 230 DRAW -B,C: DRAW A,B
 240 PLOT 125+B-Y,120-Y-B: DRAW
Y-B,0: RETURN
```

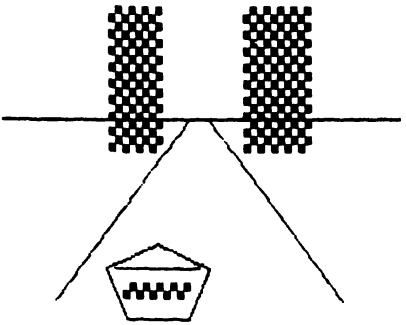


Рис. 7.26

```
  10 REM ТРЕУГОЛЬНИК ВРАЩАЮЩИЙСЯ
В ОКРУЖНОСТИ
  20 CIRCLE 130,80,75
  30 FOR F=0 TO 1E38 STEP PI/60
  40 LET X1=130+70*SIN (F)
  50 LET Y1=80+70*COS (F)
  60 LET X2=130+70*SIN (PI*2/3+F
)
  70 LET Y2=80+70*COS (PI*2/3+F)
  80 LET X3=130+70*SIN (PI*4/3+F
)
  90 LET Y3=80+70*COS (PI*4/3+F)
 100 FOR I=1 TO 2
 110 OVER 1: PLOT X1,Y1
 120 DRAW X2-X1,Y2-Y1: DRAW X3-X
2,Y3-Y2: DRAW X1-X3,Y1-Y3
 130 IF I=1 THEN PAUSE 20
 140 NEXT I: NEXT F
```

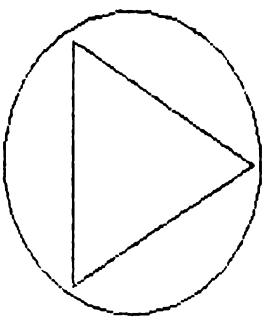


Рис. 7.27

(усложнение изображения в нужный момент и перемещение упрощенного объекта), широко применяемый в игровых программах, соответствует психологическим особенностям зрительного восприятия – до окон ли, когда на нас несется "летающая тарелка"?

Теперь рассмотрим примеры построения вращающихся на плоскости объектов. Покажем, что не обязательно составлять матрицу вращения и умножать ее на векторы всех точек. Возможны и более простые, хотя и частные решения.

Пусть требуется построить вращающийся равнобедренный треугольник. Очевидно, что все его вершины лежат на окружности, описывающей треугольник. Следовательно, для вращения треугольника достаточно перемещать три точки (вершины) по окружности, а для этого нужно задать параметр – угол с малым шагом и соединить вершины отрезками прямых (разумеется, по принципу мультипликации). Программа, реализующая этот метод, представлена на рис. 7.27.

В строке 20 строится окружность несколько большего радиуса, чем у окружности, описывающей треугольник. В строках 30–90 задается циклическое изменение угла $\varphi = F$ и формируются координаты вершин треугольника. В строке 110 треугольник то строится, то стирается (оператор OVER 1). На рисунке показано изображение треугольника, вращающегося в окружности, в случайный момент останова программы.

Как и любое другое перемещение, вращение происходит с заметными на глаз скачками.

Рассмотрим еще один пример – построение квадрата, поворачивающегося вокруг своей оси на угол α и одновременно уменьшающегося (вращение с масштабированием) [34]. Каждая сторона квадрата – отрезок прямой. Воспользуемся несколько иным, чем ранее, представлением такого отрезка. Пусть $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$ – его концевые точки. Тогда любая точка P на прямой, проходящей через эти точки, задается векторной комбинацией $(1-\mu)P_1 + \mu P_2$ для фиксированного μ, т.е. вектором

$$((1-\mu)x_1 + \mu x_2,\ (1-\mu)y_1 + \mu y_2),$$

где $\mu = \dfrac{\text{Расстояние от } P_1 \text{ до } P}{\text{Расстояние от } P_2 \text{ до } P_1}$.

Если соблюдается условие $0 \leqslant \mu \leqslant 1$, то точка лежит между точками P_1 и P_2 прямой, если $\mu < 0$, то за пределами отрезка со стороны точки P_1, если $\mu > 1$, то за пределами отрезка со стороны P_2.

Теперь вернемся к квадрату. Пусть он имеет 4 угла с координатами (x_i, y_i), где $i = 1, 2, 3, 4$, Тогда i-я сторона является отрезком прямой, соединяющей (x_i, y_i) с (x_{i+1}, y_{i+1}). При этом точка 4 соединяется с точкой 1. Произвольная точка (x_i', y_i') на i стороне квадрата может быть представлена (как показано выше) следующим образом:

$$((1-\mu)x_i + \mu x_{i+1},\ (1-\mu)y_i + \mu y_{i+1}), \tag{7.2}$$

где $0 \leqslant \mu \leqslant 1$. Величина $\mu/(1-\mu)$ является отношением, в котором рассекается заданная сторона квадрата. Если μ фиксируется, а 4 точки (x_i', y_i') нового внутреннего квадрата вычисляются указанным выше способом, то стороны нового (внутреннего) квадрата образуют угол $\alpha = \operatorname{arctg}[\mu/(1-\mu)]$ с соответствующими сторонами внешнего квадрата.

Таким образом, при фиксированном μ для каждого квадрата угол между сторонами последовательно формируемых квадратов остается неизменным и равным α. Если $\mu = 0{,}1$, то координаты каждого нового угла находятся из (7.2) как

$$(x_i', y_i') = ((9x_i + x_{i+1})/10,\ (9y_i + y_{i+1})/10). \tag{7.3}$$

Если вычерчивается $n+1$ квадрат, то внутренний квадрат повернется на угол $n\alpha$ рад относительно внешнего квадрата. Этот угол должен быть t-кратным углу $\pi/4$, т.е. $n\alpha = t\,(\pi/4)$. Из этого условия можно найти

$$\mu = \left\{\operatorname{tg}[t\,(\pi/4n)]\right\} / \operatorname{tg}[t(\pi/4n)] + 1\Big\}. \tag{7.4}$$

Итак, чтобы построить $n+1$ квадрат по заданным условиям, достаточно один раз вычислить μ по (7.4), а затем определять угловые точки каждого нового квадрата по формулам (7.3), содержащим элементарные арифметические операции. Это резко уменьшает время построения. Далее, найдя эти точки, их необходимо соединить отрезками прямых. Если $t = 3$ и $n = 21$ (строится 22 квадрата), то $\mu \approx 0{,}1$ (точнее 0,09674). Описанное построение реализует программа на рис. 7.28.

Сначала задается массив значений $Z(I) = x_i$ и $T(I) = y_i$ главного внешнего квадрата (строки 20–50). Затем организуется цикл построения 22 квадратов (строка 55). В строках 60 и 70 происходит обмен значениями переменных – переменные X(I) получают значения Z(I), а Y(I) = T(I). Далее (строки 80–110) с помощью операторов PLOT и DRAW строится квадрат. В строках 120–150 по формуле (7.3) вычисляются новые значения $Z(I) = x_i'$ и $T(I) = y_i'$. Полученная фигура имеет довольно сложный и затейливый вид, однако строится она быстро, поскольку пересчет координат угловых точек осуществляется по простым формулам, а вся фигура строится из отрезков прямых.

```
  10 REM ФИГУРА ИЗ КВАДРАТОВ
  20 DIM X(4): DIM Y(4): DIM Z(4
): DIM T(4)
  30 DATA 40,3,40,173,210,173,21
0,3
  40 FOR I=1 TO 4: READ Z(I)
  50 READ T(I): NEXT I
  55 FOR N=1 TO 22
  60 FOR I=1 TO 4: LET X(I)=Z(I)
  70 LET Y(I)=T(I): NEXT I
  80 PLOT X(1),Y(1): DRAW X(2)-X
(1),Y(2)-Y(1)
  90 DRAW X(3)-X(2),Y(3)-Y(2)
 100 DRAW X(4)-X(3),Y(4)-Y(3)
 110 DRAW X(1)-X(4),Y(1)-Y(4)
 120 FOR I=1 TO 3: LET Z(I)=(9*X
(I)+X(I+1))/10
 130 LET T(I)=(9*Y(I)+Y(I+1))/10
: NEXT I
 140 LET Z(4)=(9*X(4)+X(1))/10
 150 LET T(4)=(9*Y(4)+Y(1))/10:
NEXT N
```

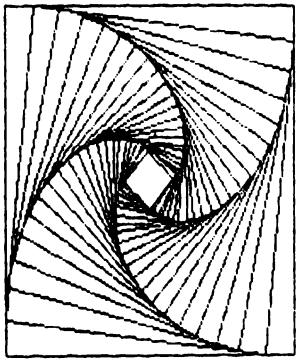


Рис. 7.28

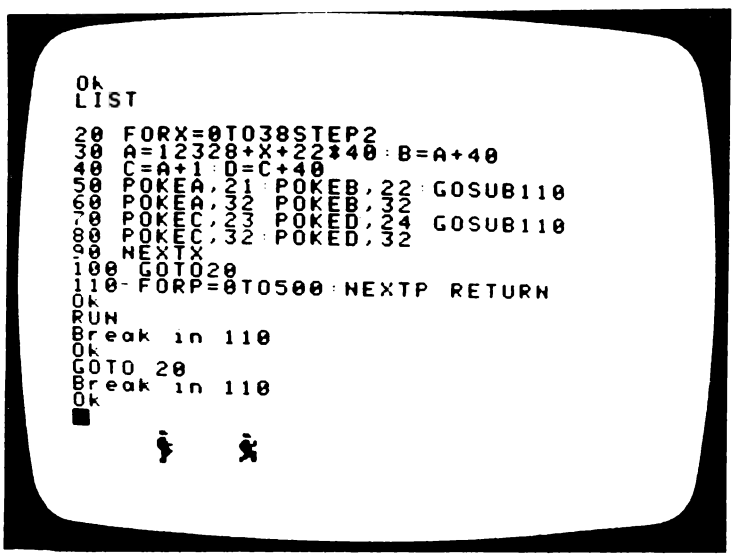


Рис. 7.29

Из всего описанного у читателя может сложиться впечатление, что построение сложных движущихся объектов (например, фигуры бегущего человека) – дело очень сложное и почти безнадежное. Это так, если пытаться строить сложные и неупорядоченные фигуры только методами машинной графики, применяя не очень-то быстродействующие ПЭВМ.

Однако есть и другой подход, легко решающий эту проблему. Он также заимствован из техники изготовления мультипликационных фильмов и заключается в том, что последовательно отображаемые объекты рисуются заранее и при съемке каждого кадра помещаются на новое место. Современные ПЭВМ имеют обширный набор графем или во всяком случае поз-

воляют пользователю составлять свои графемы, каждая может быть частью рисунка. Например, всего 4 графем достаточно, чтобы изобразить два разных кадра бегущего человека. Перемещая их, например, справа налево, мы получим картину бегущего человечка. Быстродействие ПЭВМ и в этом случае требуется достаточно высокое. Поэтому вывод графем (в тех ПЭВМ, где это возможно) лучше обеспечивать прямым вводом их кода в соответствующую ячейку ОЗУ дисплея с помощью оператора POKE (Адрес, Код). Именно на этом принципе построена программа (рис. 7.29), поясняющая построение изображения бегущего человека с помощью 5 графем.

Стирание каждого предшествующего изображения здесь задается вводом пустой графемы. Перемещение фигуры задается изменением аргумента X в цикле (строка 20). В строках 30 и 40 формируются текущие адреса. При этом используется формула (для ПЭВМ Aquarius)

$$A = 12328 + 40R + C,$$

где $R = X$ – номер позиции в столбце ($R = 0, 1, 2, \ldots, 23$) и C – номер позиции в строке ($C = 0, 1, 2, 3, \ldots, 39$). Эта формула преобразует одномерный массив адресов A в двухмерную матрицу (24×40) изображения. В строке 50 строится одна фигура (сочетанием графем с кодами 21 и 22). Обращение к подпрограмме (строка 110) задает паузу. Затем эта фигура замещается пустыми графемами с кодом 32 (строка 60), а в строке 70 строится вторая фигура с помощью графем с кодами 23 и 24 и смещенным на +1 адресом. В строке 80 она стирается. После этого (строка 90) цикл повторяется. При первом пуске директивой PRINT CHR$ (11) очищается экран дисплея. После завершения бега он повторяется (см. строку 100 с оператором возврата к строке 20).

Понимание и разумное использование описанных принципов построения движущихся объектов позволит заинтересованному пользователю ПЭВМ создавать собственные машинные фильмы. Они очень полезны в обучающих и игровых программах.

7.7. УСЕЧЕНИЕ, ПОКРЫТИЕ И СПЕЦИАЛЬНАЯ ЗАКРАСКА ФИГУР

Усечение. Усечением называется ограничение пространства, в пределах которого может находиться фигура [34]. Части фигуры, выходящие за пределы разрешенного пространства, усекаются (рис. 7.30). Частным случаем усечения является ограничение графического изображения пространством экрана дисплея или пространством страницы. В подобных случаях усечение выполняется с помощью операторов IF – THEN, ограничивающих координаты точек. Например, чтобы координата y точки A не выходила за пределы значения $y = 150$, достаточно после вычисления этой координаты записать выражение

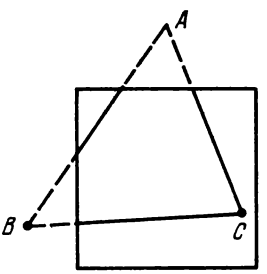


Рис. 7.30

```
[HC]   IF Y>=15Ø THEN LET Y=15Ø
```

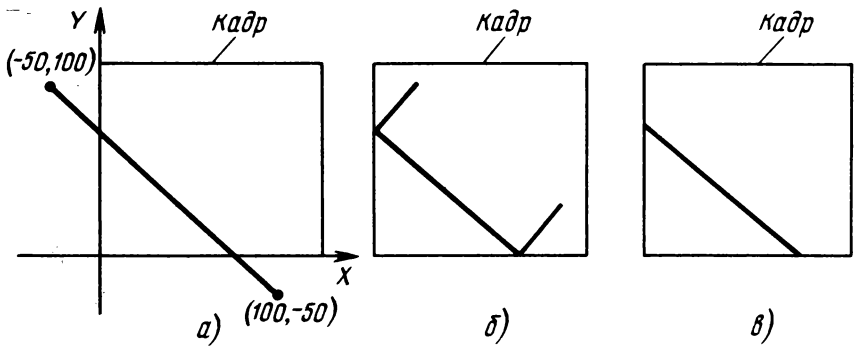


Рис. 7.31

Особые ситуации возникают, если текущие координаты x и y могут принимать отрицательные значения. Допустим, требуется усечь кадром прямую, заданную координатами двух точек (–50, 100) и (100, –50) (рис. 7.31,*а*). Обычно ПЭВМ воспринимает отрицательные значения x и y двумя способами: останавливает работу и выдает сообщение об ошибке аргумента или воспринимает x и y по модулю. В последнем случае получаются отраженные от осей X и Y отрезки прямой (рис. 7.31,*б*), которые в действительности должны усекаться. Однако, если перед заданием координат x и y в графические операторы задать условия

```
[HC1]  IF X<=0 THEN LET X=∅
[HC2]  IF Y<=0 THEN LET Y=∅
```

то будет обеспечено усечение отрезка прямой так, как это необходимо (рис. 7.31,*в*).

Разумеется, ограничивающие условия могут содержать не константы, а произвольные функции. Это обеспечивает усечение более сложными фигурами.

Покрытие. Покрытием называется заслонение одной фигуры другой, например треугольника квадратом (рис. 7.32). Операция покрытия на ПЭВМ легко обеспечивается построением одной закрашенной фигуры после другой (которая заслоняется). Таким образом, она фактически сводится к закраске фигур.

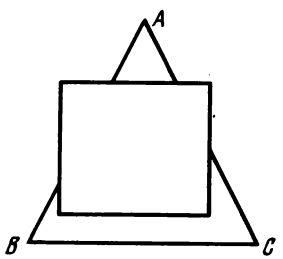


Рис. 7.32

Закраска фигур. Закраска – процесс представления фигуры или ее части в определенном цвете. При этом все точки фигуры или закрашенной ее части должны задаваться в определенном цвете. Существуют три основных способа закраски: по точкам, знакоместам и по линиям. Закраска по точкам выполняется медленно, поскольку число точек (пикселей) в фигуре может быть значительным. Этот способ применяется, когда границы области закраски произвольны (см. ниже).

```
  10 REM ПОСТРОЕНИЕ ЧАСТИЧНО ЗАК
РАШЕННОГО ПРЯМОУГОЛЬНИКА
  20 INK 0: PLOT 10,10: DRAW 0,8
0
  30 DRAW 200,0: DRAW 0,-80
  40 DRAW -200,0
  50 FOR Y=10 TO 90
  60 PLOT 11,Y: DRAW 99,0
  70 NEXT Y
```

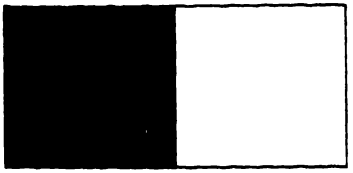


Рис. 7.33

```
   5 REM ЗАКРАСКА ЭЛЛИПСА
  10 DIM S(90): DIM C(90)
  20 FOR I=1 TO 90
  30 LET Q=I*PI/180: LET S(I)=SI
N (Q): LET C(I)=COS (Q): NEXT I
  40 FOR I=1 TO 90: LET X=80*S(I
): LET Y=40*C(I)
  50 PLOT 130+X,42+Y: DRAW -2*X,
0: PLOT 130+X,42-Y: DRAW -2*X,0
  60 NEXT I
```

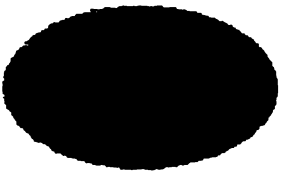


Рис. 7.34

У большинства ПЭВМ нельзя устанавливать произвольно цвет любой точки: одинаково закрашиваются все точки одного знакоместа. Чтобы ограничить необходимую емкость ОЗУ, кодируется цвет группы точек в пределах знакоместа экрана и цвет самого знакоместа. Поэтому в ряде случаев закраску фигур удобно выполнять, закрашивая знакоместа. К сожалению, в этом случае границы закраски будут резкими лишь для прямоугольников. Для других фигур они будут ступенчатыми.

Если фигура или ее часть строится из отрезков прямых, то и закраска выполняется по линиям. На рис. 7.33 представлена программа построения прямоугольника (строка 20–40), часть которого (строки 50–70) закрашена 81 горизонтальной линией. На рис. 7.34 приведена программа построения закрашенного эллипса. Вначале (строки 10–30) создается массив синусов и косинусов, необходимых для построения эллипса по его параметрическим уравнениям. Затем (строки 40–60) строится эллипс по заданным координатам точек фигуры справа и отрезки прямых длиной $2x$ (где x – координата точки относительно центра фигуры), выводимые вправо.

Закраска сложных фигур может производиться перемещением прямой по образующей кривой. На рис. 7.35 показана программа построения красного флага СССР путем перемещения вертикального отрезка прямой (строки 120–140) по образующей – слегка наклоненной вниз синусоиде (она строится в строке 130). Остальная часть программы строит древко, звезду, серп и молот. Получается картина развивающегося на ветру флага.

Интересно построение картин и смещением образующих функциональных линий. В этом случае густота закраски зависит от шага смещения и кривизны функциональной линии. Может быть получен эффект "муара". Примером является программа построения аксонометрического изображения ленты (рис. 7.36) смещением исходной синусоиды (строка 20) вправо и вверх. В этом случае закраска типа "муар" создает иллюзию освещения ленты из правого верхнего угла картины.

Нередко необходимо закрасить часть произвольной фигуры. В этом случае одна или все границы области закраски заранее неизвестны. Для опре-

```
 110 REM ФЛАГ СССР
 120 INVERSE 0: INK 2: FOR x=64
TO 184
 130 LET y=60+5*SIN ((x-60)/10)-
(x-60)/12
 140 PLOT x,y: DRAW 0,80: NEXT x
 150 INK 2: INVERSE 1: PLOT 60,1
30: DRAW 2,3
 160 DRAW 2,-3: DRAW 3,0: DRAW -
3,-2
 170 DRAW 2,-2: DRAW -4,2: DRAW
-4,-2
 180 DRAW 2,2: DRAW -3,2: DRAW 3
,0
 190 PLOT 85,122: DRAW -10,-15,-
4
 200 DRAW -4,-4: PLOT 80,118
 210 DRAW -4,-4: DRAW 1,-1: DRAW
4,4
 220 PLOT 79,115: DRAW 16,-16
 230 INK 0: INVERSE 0: FOR x=61
TO 63
 240 PLOT x,0: DRAW 0,150: NEXT
x
```

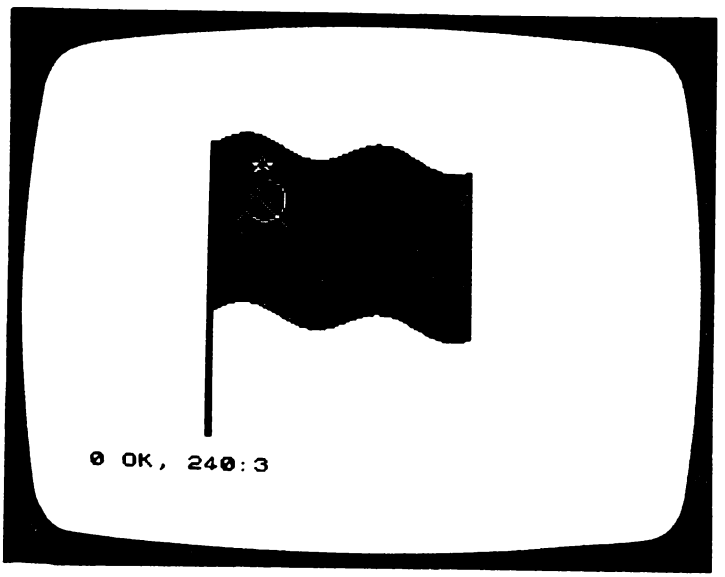


Рис. 7.35

```
10 REM ЛЕНТА
20 FOR X=0 TO 220 STEP 2
30 PLOT X,40+X/10+20*SIN (X/10
40 DRAW 20,50
50 NEXT X
```

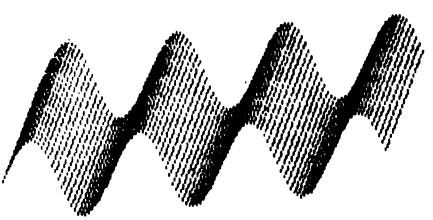


Рис. 7.36

```
 10 REM ЗАКРАСКА ЧАСТИ ФИГУРЫ С
ПОМОЩЬЮ ОПЕРАТОРА POINT
 20 CIRCLE 128,60,55
 30 FOR X=108 TO 148: LET Y=50
 40 IF POINT (X,Y)=0 THEN PLOT
X,Y: LET Y=Y+1: GO TO 40
 50 NEXT X
```

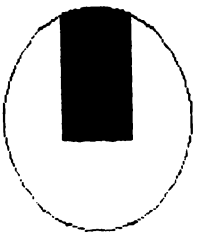


Рис. 7.37

деления границ используется функция POINT (x, y), принимающая значение 1, если точка с координатами x и y имеет цвет, отличный от цвета знакоместа ее расположения, и 0, если ее цвет тот же, что и цвет знакоместа (т.е. если точка не индицируется). На рис. 7.37 представлена программа, иллюстрирующая закраску части окружности (строка 20). Слева и справа область закраски задается крайними значениями аргумента $x = 108$ и 148 (строка 30). Контроль области закраски сверху обеспечивается строкой 40. Если значение функции POINT $(x, y) = 0$, то выводится точка с координатами (x, y), y получает приращение +1, и строка 40 выполняется вновь. В противном случае x получает приращение (оператор NEXT X в строке 50), и формируется новая вертикальная линия закраски. Вместо окружности область окраски в данном случае может быть задана любой фигурой или линией, в том числе нарисованной пользователем с помощью программы "Графический редактор" (см. § 7.10).

Преимущество последнего способа закрашивания заключается в том, что область закраски может быть и динамической. Функция POINT (x, y) полезна и в том случае, когда необходимо зафиксировать соприкосновение движущейся точки с движущимся объектом. Это часто встречается в игровых задачах типа "Стрельба по движущейся мишени".

7.8. СОЗДАНИЕ МОЗАИЧНЫХ ИЗОБРАЖЕНИЙ И КАЛЕЙДОСКОПОВ

Мозаика. Мозаика – неподвижный сложный узор из отдельных упорядоченных или неупорядоченных графических элементов. ПЭВМ открывает широкие возможности в создании мозаичных изображений.

Мозаика может быть составлена из графем ПЭВМ или сложных фигур, которые строятся с помощью подпрограммы (рис. 7.38).

Калейдоскоп. Разновидность мозаичного изображения, меняющего свой вид, обычно называется калейдоскопом. Простейший калейдоскоп – случайные точки (рис. 7.39) – часто используется в играх. Например, таким образом можно создать звездное небо в игровых "космических войнах". В данном случае координаты x и y точек задаются генератором случайных чисел и приводятся в допустимые диапазоны их изменения.

Еще один вид мозаичного калейдоскопа из знаков, которыми располагает ПЭВМ, нашел даже практическое применение – кто не видел юношей и девушек, на рубашке которых изображено нечто подобное? Программа, формирующая знаковый калейдоскоп, очень проста (рис. 7.40). В ней задается цикл изменения вспомогательной переменной (строка 20) – число

```
  10 REM МОЗАИКА ИЗ ГРАФИЧЕСКИХ
ЭЛЕМЕНТОВ СТРОЯЩИХСЯ С ПОМОЩЬЮ
ПОДПРОГРАММЫ
  20 FOR Y=20 TO 140 STEP 20: FO
R X=20 TO 220 STEP 20
  30 GO SUB 80: NEXT X: NEXT Y
  40 FOR Y=10 TO 150 STEP 20: PL
OT 10,Y
  50 DRAW 220,0: NEXT Y
  60 FOR X=10 TO 230 STEP 20: PL
OT X,10
  70 DRAW 0;140: NEXT X: STOP
  80 PLOT X,Y: DRAW 0,10,2: DRAW
 0,-10,2
  90 DRAW 10,0,2: DRAW -10,0,2
 100 DRAW 0,-10,2: DRAW 0,10,2
 110 DRAW -10,0,2: DRAW 10,0,2:
RETURN
```

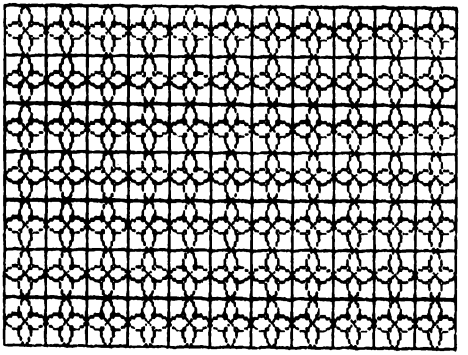


Рис. 7.38

```
10 REM СЛУЧАЙНЫЕ ТОЧКИ
20 PLOT RND*200,RND*150
30 GO TO 20
```

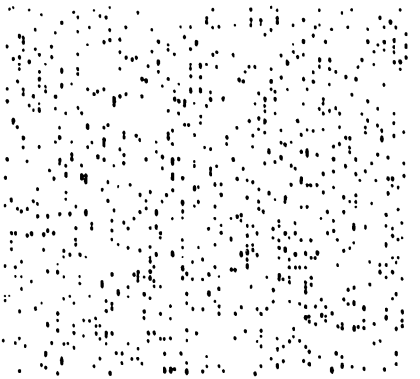


Рис. 7.39

```
  10 REM ЗНАКОВЫЙ КАЛЕЙДОСКОП
  20 FOR A=0 TO 703
  30 PRINT CHR$ (100+50*RND);
  40 POKE 23692,255: NEXT A
```

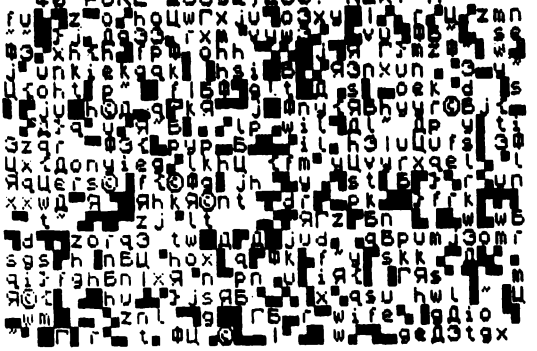


Рис. 7.40

```
  10 REM ИНТЕРФЕРЕНЦИОННЫЙ КАЛЕЙ
ДОСКОП
  20 FOR C=0 TO 7
  30 BORDER C: PAPER C: INK 7-C:
 BRIGHT 0: CLS
  40 LET S=1+2*RND
  50 FOR N=0 TO 255 STEP S
  60 PLOT N,0
  65 DRAW OVER 1;255-2*N,175
  70 NEXT N
  80 FOR N=0 TO 175 STEP S
  90 PLOT 0,N
 100 DRAW OVER 1;255,175-2*N
 110 NEXT N
 120 PAUSE 0: NEXT C: RUN
```

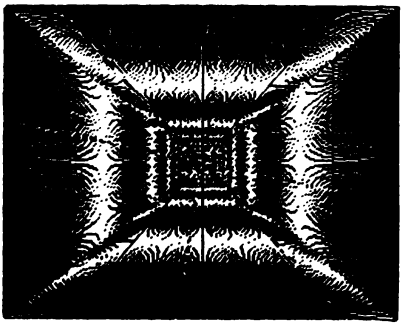


Рис. 7.41

```
  10 REM КАЛЕЙДОСКОП ИЗ ОКРУЖНОС
ТЕЙ
  20 BORDER 7: PAPER 7: BRIGHT 0
  30 FOR C=0 TO 7
  40 FOR N=2 TO 80 STEP 2+C/2
  50 CIRCLE OVER 1; INK C; PAPER
 7-C;110,87,N
  60 CIRCLE OVER 1; INK C; PAPER
 7-C;134,87,N
  90 NEXT N: NEXT C
 100 GO TO 30
```

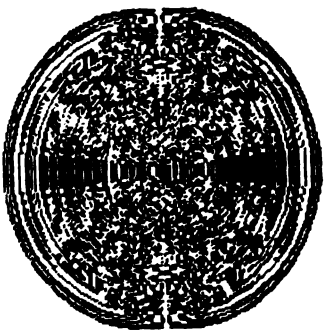


Рис. 7.42

вводимых знаков (в нашем случае их 704, т. е. A = 0, 1, . . . , 703). Оператор CHR$ (строке 30) имеет меняющийся по случайному закону код знака – аргумент (100 + 50 * RND). Это значит, что, поскольку RND находится в интервале [1, –1], то задаются коды от 50 до 150. Изменив интервал задания кодов, можно выбрать характер мозаики, например сделать ее только из цифр, только из букв, только из графем или смешанных знаков (как в нашем случае). Оператор CHR$ преобразует код в знак, а оператор POKE засылает этот знак в ОЗУ дисплея.

Таким образом, для быстрого построения мозаик и калейдоскопов целесообразно пользоваться операторами системных функций.

Особенно интересны калейдоскопы с непрерывной сменой многоцветного изображения. Они по своему воздействию очень напоминают детскую игрушку – калейдоскоп с множеством разноцветных стеклышек. Даже на экране черно-белого телевизора калейдоскопы создают причудливые и непрерывно меняющиеся изображения со множеством полутонов вместо цветов. Очень интересно наблюдать за "фантазией" ПЭВМ, строящей красочные рисунки. Польза от создания таких калейдоскопов очевидна. Они могут успешно применяться для создания художественного орнамента, рисунков для тканей, обоев и т. д.

Простые калейдоскопы можно создавать, используя эффект "муара" при построении даже простейших смещенных линий – прямых или окруж-

ностей. На рис. 7.41 приведена одна из таких программ (ПЭВМ ZX-Spectrum), по которой можно построить множество прямых просто поворотом одной прямой. Однако при этом шаг меняется по случайному закону, оставаясь достаточно малым (это обеспечивает получение эффекта "муара"). Одна из множества картин, полученных с помощью этой программы, также приведена на рисунке. Характер картин можно менять, нажимая любую клавишу (получаем пакет из 8 картин) либо изменяя данные в строке 40.

Еще более забавные картины создает калейдоскоп, в основе которого лежит построение двух систем из концентрических окружностей с мелким шагом. Каждая окружность засвечивает определенным цветом ряд знакомест экрана. В результате изображение непрерывно меняется, создавая причудливые фигуры (рис. 7.42).

Большое количество программ создания абстрактной графики (мозаики, калейдоскопов и др.) заинтересованный читатель найдет в литературе по программному обеспечению конкретных ПЭВМ.

7.9. РАЗРАБОТКА ГРАФИЧЕСКИХ РЕДАКТОРОВ – ПРОГРАММ ДЛЯ ПОСТРОЕНИЯ И РЕДАКТИРОВАНИЯ ПРОИЗВОЛЬНЫХ ГРАФИКОВ

Все приведенные программы создавали графику средствами машины (машинная графика). Часто необходимо создавать с помощью ПЭВМ произвольные изображения, задаваемые пользователем, например принципиальных схем электронных устройств, технических рисунков, художественных образов и т. д. В этом случае ПЭВМ выполняет роль карандаша или кисти и листа бумаги или холста.

Имеется множество технических возможностей для подобного режима работы. Так, ПЭВМ имеет специальное оптико-электронное устройство, по размеру и виду напоминающее авторучку, – световое перо. Приблизив его к экрану дисплея, можно в заданном месте поставить точку или взять под свое управление уже имеющуюся вблизи кончика пера точку. Эту точку можно передвигать, строя сложную кривую, или указывать точки, которые должны соединяться отрезками прямых или дуг. У некоторых ПЭВМ под экраном дисплея есть шкала цветов. Указав карандашом цвет на шкале, можно затем перенести его на точку, указанную на экране дисплея, закрасив ее или часть фигуры этим цветом.

Другой способ рисования – с помощью передвигаемого по гладкой поверхности датчика – "мыши" – реализован в новейших ПЭВМ Macintosh и Lisa фирмы Apple (США). Здесь кривая на экране дисплея повторяет движения "мыши". Манипулируя кнопками "мыши", можно указывать точки и другие объекты изображения, менять их цвет, создавать новые объекты и т. д.

Многие ПЭВМ имеют специальные клавиши со стрелками (вверх, вниз, вправо и влево), с помощью которых можно гонять точку на экране и создавать линии. Однако у ряда ПЭВМ эти клавиши означают лишь перемещение курсора. Тем не менее программным путем всегда можно создавать такую возможность.

В состав системного программного обеспечения сложных ПЭВМ часто

входит специальная программа "Графический редактор", с помощью которой перечисленными методами можно создавать и редактировать произвольные изображения. Такая программа нередко включается в библиотеку прикладных программ большинства современных типов ПЭВМ. Графический редактор обычно обеспечивает следующие возможности: задание точек с определенными координатами, передвижение ее по экрану с поднятым и опущенным "пером", изменение цвета точки, формирование перемещением точки различных линий, задание отрезков прямых – векторов, дуг и окружностей, закраска замкнутых фигур, редактирование рисунков, в частности удаление ошибочно сделанных фрагментов рисунка, и т.д. Графический редактор – это большая и довольно сложная программа, объединяющая разнообразные приемы машинной графики. Поскольку составление таких программ – процесс весьма трудоемкий, лучше пользоваться готовыми программами. Однако, как показывает практика, нередко готовые программы в чем-то не удовлетворяют пользователя, например занимают чрезмерно большую емкость ОЗУ, считывание полученных изображений идет слишком долго, программа чересчур дорога. В этом случае сносные, а главное подстроенные под свои нужды программы, пользователь вполне может составить и сам.

Примером может служить простая программа для построения графиков произвольных функций одной переменной $y(x)$ на фоне координатных осей с оцифрованными делениями (ПЭВМ ZX-Spectrum):

```
  50 CLS : PRINT AT 8,2;"ПОСТРОЕ
НИЕ ГРАФИКОВ ФУНКЦИЙ"
  80 PRINT AT 15,4;"НАЖМИТЕ ЛЮБУ
Ю КЛАВИШУ": PAUSE 0
 100 CLS : INPUT "ВВЕДИТЕ Xmin="
;xmin;" Xmax=";xmax
 120 IF xmin>=xmax THEN GO TO 10
0
 130 LET ymax=12: LET ymin=-12
 140 INPUT "ВАС УСТРАИВАЕТ ymin=
-12, ymax=12 (y/n)? ";y$
 150 IF y$="y" OR y$="Y" THEN GO
 TO 200
 160 INPUT "ВВЕДИТЕ Ymin=";ymin;
" Ymax=";ymax
 170 IF ymin>=ymax THEN GO TO 16
0
 200 REM ОСИ КООРДИНАТ
 230 LET xrange=xmax-xmin: LET y
range=ymax-ymin: LET dx=xrange/2
00: LET dy=yrange/144
 250 LET cy=ABS (ymin)*(ymin<0):
 LET cy=cy/dy+16
 260 LET cx=ABS (xmin)*(xmin<0):
 LET cx=cx/dx+32
 280 PLOT 32,cy: DRAW 200,0
 290 PLOT cx,16: DRAW 0,144
 310 LET yscale=INT ((175-cy)/8)
+1
 320 LET xscale=INT (cx/8)-4
 330 FOR n=0 TO 5
 340 LET sx=(INT (10*(xmin+n*xra
nge/5)+.5))/10
 350 PRINT OVER 1;AT yscale,5*n+
4;sx: PLOT (5*n+4)*8,cy-1: NEXT
n
 360 FOR n=0 TO 6
 370 LET sy=(INT (10*(ymax-n*yra
nge/6)+.5))/10
 380 LET g$=STR$ sy
 390 IF LEN g$<4 THEN FOR c=LEN
g$ TO 3: LET g$=" "+g$: NEXT c
```

```
 400 PRINT OVER 1;AT 3*n+1,xscal
e;g$: PLOT cx+1,(3*n+2)*8: NEXT
n
 405 INPUT "ОСИ ПОДХОДЯТ? (Y/N)
";Z$: IF Z$="N" OR Z$="n" THEN G
O TO 100
 408 INPUT "y(x)=";f$
 410 REM ГРАФИК ФУНКЦИИ f(x)
 415 LET x=xmin: LET y0=(VAL (f$
)-ymin)/dy: PLOT 32,y0+16
 420 FOR n=1 TO 199
 430 LET x=xmin+n*dx
 440 LET y=(VAL (f$)-ymin)/dy: I
F y>=158 THEN LET y=158: LET y0=
y: GO TO 460
 450 LET x1=n+32: LET y1=y: IF y
1<-15 THEN LET y1=-15: LET y0=-1
5: GO TO 460
 455 PLOT n+31,y0+16: DRAW 1,y1-
y0: LET y0=y0+INT ((Y1-Y0)+.5)
 460 NEXT n
 480 INPUT "БУДЕТЕ ВВОДИТЬ НОВЫЙ
 ГРАФИК (Y/N)? ";y$
 490 IF y$="y" OR y$="Y" THEN GO
 TO 408
```

Эта программа при пуске (строки 50 и 80) выдает сообщение

ПОСТРОЕНИЕ ГРАФИКОВ ФУНКЦИЙ
НАЖМИТЕ ЛЮБУЮ КЛАВИШУ

При нажатии любой клавиши экран очищается и в служебной строке появляется запрос (строки 100 и 120):

ВВЕДИТЕ X min =⟨Ввод⟩ X max =⟨Ввод⟩

После ввода (строка 130) устанавливаются масштабы изображения y max = 12 и y min = –12 по оси Y, после чего (строки 150–170) делается запрос

ВАС УСТРАИВАЕТ y min = –12, y max = 12 (y/n)?

Если пользователь отвечает "нет" (n), следует новый запрос

ВВЕДИТЕ y min =⟨Ввод⟩ y max =⟨Ввод⟩

В строках 200–405 строятся координатные оси, положение осей вычисляется автоматически, разметка оси делениями и проставление против них цифры. Затем (строка 405) следует запрос

ОСИ ПОДХОДЯТ (Y/N)?

Если введен ответ "нет" (n), процедура задания масштаба повторяется. При утвердительном ответе следует запрос

y (x) = " "

т.е. надо задать выражение для y (x). Оно приписывается символьной переменной f$. Допустим, было задано y (x) = "cos x". Автоматически будет построен график функции y (x) = cos (x) и выдан запрос

БУДЕТЕ ВВОДИТЬ НОВЫЙ ГРАФИК (Y/N)?

При утвердительном ответе повторяется запрос y (x) = " " и новый график можно наложить на прежний.

На рис. 7.43 показаны построенные по этой программе три графика: $y(x) = \cos x$, $y(x) = 1 - e^{-x}$ и $y(x) = (1 - e^{-x}) \sin(5x)$. Было задано $x_{min} = 0$, $x_{max} = 10$, $y_{min} = -1{,}2$ и $y_{max} = 1{,}2$.

Следующая программа рассчитана на составление технических рисунков и реализует все упомянутые выше функции графических редакторов, кроме закраски замкнутых фигур (программа П1.12). Она дает три принципиально разных подхода к созданию рисунков:

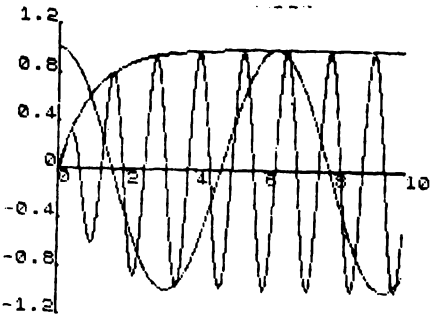


Рис. 7.43

1) генерация бейсик-программ, которая создает полную копию синтезированного рисунка;

2) создание массива данных рисунков для универсальной и простой бейсик-программы, создающей рисунки любой сложности;

3) подготовку рисунка на экране дисплея с последующим снятием его копии с помощью принтера или записью на магнитные носители (диски или кассеты).

В строках 20–110 дана универсальная бейсик-программа, создающая рисунки по массивам, заполняемым основной программой графического редактора – она начинается со строки 9000. При работе последней (см. ниже) формируются 4 одномерных массива:

x(i) = ±*nnn.p* – массив координат *x nnn* и кодов операторов *p*,

y(i) = ±*nnn.i* – массив координат *y nnn* и кодов цвета *i*,

r(i) – массив радиусов для операторов DRAW и CIRCLE,

c$ (i) – символьный массив.

Коды операторов:

1 – PLOT *x, y*

2 – DRAW *x, y, r*

3 – CIRCLE *x, y, r*

4 – PRINT TAB *y, x*; c$

Каждое элементарное действие в основной программе графического редактора (они заданы кодом *p*) сопровождается созданием одного элемента перечисленных массивов. В работе основной программы легко разобраться с помощью комментариев:

Номера строк	Комментарий
9000	Наименование программы
9005	Меню выполняемых функций
9010	Задание цвета бордюра и страницы
9030	Задание массивов
9040	Задание цвета маркера
9050	Вывод маркера (.) в исходное состояние
9055	Задание исходных данных и очистка экрана
9060	Придание клавишам ↑, ↓, →, ← функций клавиш перемещения маркера

Окончание табл.

Номера строк	Комментарий
9100	Подъем (0) и спуск (1) "пера"
9130	Выбор из "меню" операций
9260	Задание точки
9300	Задание отрезка прямой – вектора или отрезка дуги
9370	Задание окружности с запросом "Radius r = "
9410	Задание символа или строки
9450	Стирание точки
9460	Стирание отрезка прямой-вектора или отрезка дуги
9470	Стирание окружности
9480	Стирание символа или строки

После пуска программы появляется сообщение о ее возможностях и запросы на цвет бордюра, страницы и знака, а также на резервирование памяти под массив точек графика. После ответа на запросы появляется маркер – мигающая точка, управляемая клавишами ее перемещения ←, ↓, ↑ и →. Нажатие клавиши 1 включает световое перо, клавиши 0 – выключает его. Так можно рисовать различные фигуры – режим 1.

Режим 2 программы – построение графиков с помощью команд PLOT, DRAW, LINE и CIRCLE с одновременной генерацией строк бейсик-программы и запоминанием построений в массиве данных. Так, введя команду PLOT, мы фиксируем точку, а в верху экрана появляется строка вида (пример условный)

```
1  INK Ø: PLOT 1Ø, 9Ø
```

Переместив маркер в новое положение, можно построить отрезок прямой нажав клавишу L (команда LINE). Появится новая строка бейсик-программы

```
2  INK Ø: DRAW 5Ø, Ø
```

Клавиша i вводит добавочное меню команд: INK (клавиша x) – смена цвета, DRAW (клавиша w) – построение дуг или отрезков прямой по команде DRAW *x, y, r* (с запросом *r*), CIRCLE (клавиша h) – построение окружностей с центром, указанным курсором и радиусом *r* (по запросу), и p (клавиша p) – печать символа или графемы на месте маркера с предварительным обозначением всего знакоместа темным квадратом. Любая операция может быть стерта нажатием клавиши BREAK SPACE (с устранением ее строки бейсик-программы). Передвижение маркера на один шаг сопровождается звуком в виде щелчка.

Редактор не вводит в ОЗУ бейсик-программу, а лишь подсказывает ее пользователю. Если ввести ее, она просто повторит рисунок, который создал пользователь. Этот режим работы наиболее полезен при построении простых графиков и занимает минимальную емкость ОЗУ. При выполне-

нии режима 2 вся информация о построении графиков фиксируется в массивах (см. выше).

Режим 3 – построение графиков с помощью универсальной бейсик-программы. Массивы можно записать на магнитный накопитель. На рис. 7.44 показана построенная редактором схема каскада с операционным усилителем. Следует отметить, что график можно записать на магнитную ленту и воспользоваться командой SAVE "Имя" SCREEN$.

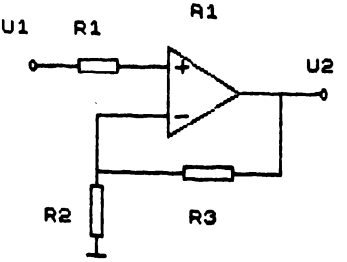


Рис. 7.44

Универсальная бейсик-программа очень проста и содержит всего 90 строк (строки 20–110). Несмотря на это, она способна строить рисунки любой сложности. Память ОЗУ используется лишь для хранения массивов (не следует забывать о том, что они должны быть записаны на магнитный носитель по завершении работы с графическим редактором). Массивы можно записать совместно с универсальной бейсик-программой, уничтожив ставший ненужным редактор.

Универсальная бейсик-программа с помощью операторов IF – THEN – Инструкция обеспечивает поэлементное выполнение соответствующей операции. Определенную трудность представляет задание размерности массива, ибо она заранее точно неизвестна. Например, для рис. 7.44 требуется задание размерности DIM = $n = 5\emptyset$. Рекомендуется задавать массив с определенным запасом.

Принципы, положенные в основу описанной программы, характерны и для других программ, реализованных на ПЭВМ другого типа. Поэтому пользователь может составить подобную программу и для других ПЭВМ.

7.10. АППРОКСИМАЦИЯ И ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ НА ПЛОСКОСТИ ФУНКЦИЙ ДВУХ ПЕРЕМЕННЫХ

В практических приложениях часто встречаются функции двух переменных $z(x, y)$, представляемые в плоскости X, Z в виде семейства кривых $z(x)$ при ряде фиксированных значений y. Такое представление имеют, например, выходные вольт-амперные характеристики биполярных и полевых транзисторов, в таком виде эти функции приводятся в справочниках. Встречается и задание их в виде таблиц. Между тем в ряде расчетов бывает необходимо определять функции $z(x, y)$ при произвольных значениях как x, так и y. Поэтому приходится искать подходящую аппроксимацию.

В некоторых случаях для зависимости $z(x)$ при $y = y_0$ удается найти простую аналитическую аппроксимацию. Тогда различные семейства кривых $z(x, y)$ могут быть получены с помощью преобразований графических изображений: изменение масштаба по осям Z и X, смещения осей и т. д.

Рассмотрим, например, зависимость $z = y_0(1 - e^{-x})$ при $y_0 = 1$ (рис. 7.45,*а*). Изменяя y_0 (т. е. меняя масштаб графика по оси Z), получаем

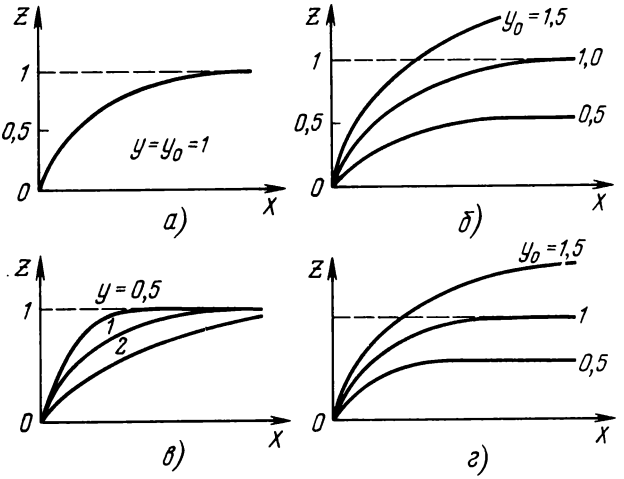


Рис. 7.45

семейство кривых (рис. 7.45,*б*). Теперь рассмотрим функцию $z = (1 - e^{-x/y_0})$ при $y_0 = 1$, аналогичную только что приведенной. Если в ней изменять y_0, изменяется масштаб графика по оси X и получается новое семейство кривых (рис. 7.45,*в*). В функции $z = y_0(1 - e^{-x/y_0})$ изменение y_0 ведет к одновременному изменению масштаба графика и по оси Z, и по оси X. При этом семейство кривых имеет вид, приведенный на рис. 7.45,*г*.

В данном случае изменение масштаба графиков при изменении y_0 линейное, но оно может быть и нелинейным. Этот прием аппроксимации успешно используется во многих технических приложениях. Например, семейство выходных вольт-амперных характеристик мощных МДП-транзисторов с горизонтальным каналом хорошо аппроксимируется выражением

$$I_C(U_З, U_C) = S(U_З - U_{отс} - bU_З^2)\{1 - \exp[-kU_C/(U_З - U_{отс} - bU_З^2)]\}.$$

Здесь ток стока $I_C = z$ – функция напряжения на строке $U_C = x$ и напряжения на затворе $U_З = y$ (относительно истока); S – крутизна транзистора; b и k – параметры аппроксимации; $U_{отс}$ – напряжение отсечки (при $U_З < U_{отс}$ следует полагать $I_C = 0$). Заметим, что здесь зависимость $I_C(U_З) = S(U_З - U_{отс} - bU_З^2)$ есть передаточная характеристика мощного МДП-транзистора. Она имеет вид параболической кривой с протяженным линейным участком и загибом вверху при больших $U_З$, что учитывается членом $bU_З^2$.

Из этого примера видно, что подобная аппроксимация требует введения ряда подгоночных параметров, в нашем случае их четыре (S, $U_{отс}$, b и k). Некоторые из них (b и k), как и вся аппроксимация, являются формальными, т.е. не вытекают из точного теоретического расчета зависимости $I_C(U_C, U_З)$. Это само по себе не мешает эффективно использовать подоб-

ные аппроксимации, так как нередки случаи, когда точное аналитическое описание функций двух переменных весьма сложно или оно отсутствует вообще. Тем не менее описанные приемы носят частный характер. Они полезны в тех случаях, когда максимальная погрешность аппроксимации 10 – 20% допустима и когда вычисления значений функции нужно выполнять в максимально возможное время.

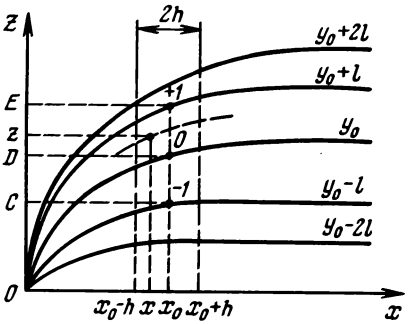


Рис. 7.46

Рассмотрим более общий прием аппроксимации функций $z(x, y)$, не требующий ввода никаких дополнительных формальных параметров. Этот прием заключается в двухмерной локальной квадратической многоинтервальной аппроксимации функции $z(x, y)$.

Будем считать функцию $z(x, y)$ заданной в виде семейства кривых $z(x)$ при заданных y или таблицей значений $z(x)$ при ряде y. Допустим, что график зависимости $z(x, y)$ имеет вид, приведенный на рис. 7.46. Чтобы найти $z(x, y)$, при $x \approx x_0$ и $y \approx y_0$, вначале выберем три кривые, у которых значения y, ближайшие к y_0. Выделим интервал $2h$ аппроксимации, в пределах которого лежит значение x. Теперь можно найти значение z для точек -1 и $+1$, лежащих на выбранных кривых, используя явную формулу квадратичной интерполяции Лагранжа. Таким образом получаем три значения z: $z(x_0, y_0 - l)$, $z(x_0, y_0)$ и $z(x_0, y_0 + l)$, где l – интервал интерполяции по переменной y. Затем можно интерполировать по переменной y и найти окончательно $z(x, y)$.

Алгоритм двухмерной аппроксимации-интерполяции следующий [1].

1. Функция $F = z(x, y)$ задается в виде матрицы $F(I, J)$, где I – номер строки таблицы (или кривой графика); J – номер столбца. При нумерации I и J с нуля $0 \leqslant I \leqslant N-1$ и $0 \leqslant J \leqslant M-1$, где N и M – число строк и столбцов. Задаем также приращения (шаги) $\Delta x = h = \text{const}$ и $\Delta y = l = \text{const}$, а также граничные значения a для x и b для y.

2. Для каждого заданного x и y находим

$J = \text{int}((x-a)/h)$ при $J \neq 0$, $J = 1$, если получим $J = 0$;
$I = \text{int}((y-b)/l)$ при $I \neq 0$, $I = 1$, если получим $I = 0$;
$P = (x - a - Jh)$; $Q = (y - b - Il)$.

3. Для $I = I - 1$, $I = I$ и $I = I + 1$ (в правой части I, вычисленное в п.2), применяя явную интерполяционную формулу Лагранжа для квадратичной интерполяции трижды, вычисляем значение z:

$$z = P(P-1)F(I, J-1)/2 + (1-P^2)F(I, J) + P(P+1)F(I, J+1)/2.$$

Заметим, что это соответствует получению точек -1, 0 и $+1$ на рис. 7.46. Значения z присвоим переменным C, D и E.

4. Выполняем квадратичную интерполяцию по переменной y с помощью формулы $z = Q(Q-1)C/2 + (1-Q^2)D + Q(Q+1)E/2$.

Реализующая этот алгоритм программа:

```
  10 PRINT "МНОГОИНТЕРВАЛЬНАЯ
 ДВУХМЕРНАЯ"," АППРОКСИМАЦИЯ-ИН
ТЕРПОЛЯЦИЯ","      ФУНКЦИИ Z=F(X
,Y)"
  20 INPUT "ЗАДАЙТЕ ЧИСЛО ИНТЕРВ
АЛОВ ПО ОСИ X M=";M
  30 INPUT "ЗАДАЙТЕ ЧИСЛО КРИВЫХ
 N=";N: DIM F(N,M+1)
  40 INPUT "ЗАДАЙТЕ X0,Y0 ";A,B
  50 INPUT "ЗАДАЙТЕ DELTA X,DELT
A Y ";H,L
  60 LET F1=0: FOR I=1 TO N: FOR
 J=1 TO M+1
  70 PRINT "F";I-1;",";J-1;"=";
 INPUT "ВВЕДИТЕ F(I,J)=";F(I,J):
 PRINT F(I,J)
  80 IF F1<F(I,J) THEN LET F1=F(
I,J)
  90 NEXT J: NEXT I
 100 CLS : PRINT AT 4,3;"МЕНЮ ОП
ЕРАЦИЙ:"
 110 PRINT AT 4,3;"КОД 0-ВЫЧИСЛЕ
НИЕ Z=F(X,Y)": PRINT AT 5,3;"КОД
 1-ВЫВОД ГРАФИКА F(X,Y)": PRINT
AT 6,3;"КОД 2-ДРУГИЕ ОПЕРАЦИИ"
 120 INPUT "ВВЕДИТЕ КОД ";K: IF
K=0 THEN GO TO 160
 130 IF K=1 THEN GO TO 300
 140 IF K=2 THEN GO TO 500
 150 GO TO 100
 160 INPUT "ВВЕДИТЕ Y,X ";Y,X: G
O SUB 200
 170 CLS : PRINT "X=";X,"Y=";Y:
PRINT "F(X,Y)=";Z: PRINT "НАЖМИТ
Е ЛЮБУЮ КЛАВИШУ": PAUSE 0: GO TO
 100
 200 REM ВЫЧИСЛЕНИЕ Z=F(X,Y)
 210 LET J=INT ((X-A)/H): IF J<=
0 THEN LET J=1
 220 LET I=INT ((Y-B)/L): IF I<=
0 THEN LET I=1
 230 LET P=(X-A-J*H)/H: LET Q=(Y
-B-I*L)/L: LET P2=P*P: LET Q2=Q*
Q
 240 LET I=I-1: GO SUB 280: LET
C=Z
 250 LET I=I+2: GO SUB 280: LET
E=Z
 260 LET I=I-1: GO SUB 280: LET
D=Z
 270 LET Z=((Q2-Q)*C+(Q2+Q)*E)/2
+(1-Q2)*D: RETURN
 280 LET Z=((P2-P)*F(I+1,J)+(P2+
P)*F(I+1,J+2))/2+(1-P2)*F(I+1,J+
1): RETURN
 290 RETURN
 300 CLS : GO SUB 9600: PRINT AT
 0,3;"MY=";F1: INPUT "ВВЕДИТЕ МА
СШТАБ MX=";X3: PRINT AT 0,15;"MX
=";X3
 305 PRINT AT 0,29;"Ú3";AT 21,29
;"UC";AT 1,0;"IC"
 310 INPUT "ВВЕДИТЕ MGRAF=";W
 320 LET X2=200/W: FOR K=1 TO N:
 LET Y1=150*F(K,1)/F1: PLOT 30,1
50+Y1
 330 FOR T=2 TO W+1: LET X=A+T*X
3/(W+1): IF T=W+1 THEN LET X=X-.
00001
 335 LET Y=B+(K-1)*L: LET Y2=Y:
IF K=N THEN LET Y=Y-.00001
 340 GO SUB 200: LET Y=150*Z/F1
 350 DRAW X2,Y-Y1: LET Y1=Y: NEX
T T: PRINT AT 21-(Y1+10)/8,29;Y2
: NEXT K: PAUSE 0: GO TO 100
 500 GO TO 100
```

```
9600 REM ПОСТРОЕНИЕ КООРДИНАТНЫХ
 ОСЕЙ
9605 PLOT 30,160: DRAW 0,-150: D
RAW 200,0
9610 FOR O=50 TO 230 STEP 20
9615 PLOT O,10: DRAW 0,4: NEXT O
9620 FOR O=25 TO 160 STEP 15
9625 PLOT 30,O: DRAW 4,0: NEXT O
9630 PRINT AT 1,2;"1": PRINT AT
11,1;".5": PRINT AT 21,2;"0"
9635 PRINT AT 21,15;".5": PRINT
AT 21,28;"1": RETURN
```

В этой программе x_0 = A и y_0 = B – нижние границы допустимых значений x и y.

В строках 20–90 вводятся значения M и N, Δx = H, Δy = L и таблица F (I, J). В строках 100–170 задается код операций: 0 – вычисление $Z = F(x, y)$ при заданных x и y, 1 – вывод графика $z = F(x, y)$ и 2 – другие операции (этот код вводится при использовании данной программы совместно с другими программами, которые записываются со строки 500). В строках 200 – 290 реализован описанный выше алгоритм вычисления $z = F(x, y)$, в строках 300–350 вывод графиков $z = F(x, y)$ при заданном числе точек каждого графика MGRAF. При построении графика каждой кривой используется линейная интерполяция с помощью оператора DRAW. Не рекомендуется задавать MGRAF менее 15–20, в противном случае кривая будет явно выглядеть как ломаная, состоящая из отрезков прямых. В строках 9600 – 9635 – подпрограмма построения координатных осей и поясняющих надписей. Предусмотрено автоматическое масштабирование графика по оси Z и вывод значений $y = U_З$ (в данном случае строится график выходных характеристик мощного полевого транзистора) справа от конца каждой кривой.

Работа с программой протекает следующим образом. После вывода названия программы по запросам ПЭВМ вводятся M, N, XØ= x_0, YØ= y_0, Δx = = H, Δy = L и таблица $z = F(x, y)$. Допустим M = 5, N = 6, H = 10 В, L = 1 В, XØ = 0 и YØ = $U_{отс}$ = 3 В. Таблица z (x, y) имеет вид

y \ x	0 В	10 В	20 В	30 В	40 В	50 В
3 В	0	0	0	0	0	0
4 В	0	0,35	0,4	0,45	0,5	0,55
5 В	0	0,8	1,1	1,2	1,22	1,25
6 В	0	1,25	1,8	2,08	2,12	2,15
7 В	0	1,8	2,65	3,1	3,22	3,28
8 В	0	2,85	4,05	4,45	4,62	4,65

Введя эти данные, получим на экране дисплея сообщение

МЕНЮ ОПЕРАЦИЙ
КОД Ø ВЫЧИСЛЕНИЕ Z = F (X, Y)
КОД 1 ВЫВОД ГРАФИКА F (X, Y)
КОД 2 ДРУГИЕ ОПЕРАЦИИ
ВВЕДИТЕ КОД

Если ввести код 1, ПЭВМ дает запрос

ВВЕДИТЕ МАСШТАБ MX =

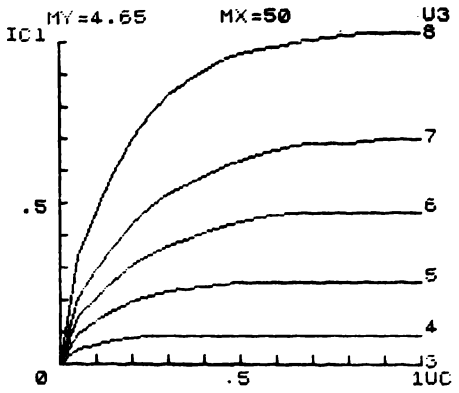


Рис. 7.47

Вводим MX = 50, так как верхнее деление горизонтальной оси графика помечено цифрой 1. После ввода по запросу

ВВЕДИТЕ MGRAF =

числа точек графика MGRAF = 20 ПЭВМ выдает график семейства выходных вольт-амперных характеристик мощного МДП-транзистора (рис. 7.47), заданного приведенной выше таблицей.

Подпрограмма вычисления $z = F(x, y)$ (строка 200 – 290) может использоваться в составе других программ, например моделирования каскада на мощном МДП-транзисторе (см. § 7.12). В этом случае задается код 2, со строки 500 должна вводиться дополнительная программа. Эта программа способна аппроксимировать семейства сложных кривых, не обязательно монотонных, но однозначных при задании аргументов x и y. Примером могут служить вольт-амперные характеристики транзисторов и различных схем с Λ- и N-образными вольт-амперными характеристиками. Еще раз подчеркнем такое достоинство программы, как аппроксимация $z(x, y)$ без введения каких-либо подгоночных коэффициентов. Уже при числе точек каждой кривой около 5 – 6 (см. пример выше) достигаются высокая точность аппроксимации и плавность построенных кривых. Заметим, что кривые точно проходят через узловые точки. Первые производные в них испытывают скачки, как правило, незаметные, так как квадратичная аппроксимация на малом интервале $\pm h$ дает достаточно высокую точность.

Следует отметить, что вычисление $z(x, y)$ при заданных x и y требует четырехкратного обращения к формуле Лагранжа. Затраты времени можно сократить, комбинируя представление $z(x)$ в виде одномерного сплайна с преобразованиями масштаба, описанными ранее (см. § 7.4). При расчетах на быстродействующих ПЭВМ описанный алгоритм двухмерной аппроксимации можно использовать совместно с явными интерполяционными формулами Лагранжа более высокого порядка (такие формулы вплоть до 6-го порядка приведены в [1, 29 – 33]).

7.11. РАБОТА С ТИПОВЫМИ ПРОГРАММАМИ ГРАФИКИ

Типовые штатные программы графики входят в прикладное программное обеспечение всех современных ПЭВМ. Как правило, это довольно сложные и универсальные программы, рассчитанные на применение их пользователями в готовом виде с вводом в ПЭВМ с магнитных носителей. У ряда ПЭВМ такие программы встроены в ПЗУ, т. е. составляют часть системного программного обеспечения. На их долю приходится до 70–80% задач, связанных с практическим массовым использованием графики

ПЭВМ. Поэтому пользователю важно научиться использовать такие программы.

Универсальная программа построения графиков UGD (см. программу П1.6). При пуске ее на экране дисплея ПЭВМ появляется меню из 8 основных операций:

```
          ***UGD***
  УНИВЕРСАЛЬНАЯ ПРОГРАММА
    ПОСТРОЕНИЯ ГРАФИКОВ

          МЕНЮ  f(X):

1-АНАЛИТИЧЕСКАЯ
2-ЗАДАЕТСЯ ПОДПРОГРАММОЙ
3-АППРОКСИМИРУЕТСЯ ПОЛИНОМОМ
4-АППРОКСИМИРУЕТСЯ ПОЛИНОМОМ
  МЕТОДОМ НАИМЕНЬШИХ КВАДРАТОВ
5-СПЛАЙН-АППРОКСИМАЦИЯ ГЛОБАЛ.
6-СПЛАЙН-АППРОКСИМАЦИЯ ЛОКАЛ.
7-МНОГОИНТЕРВАЛЬНАЯ КВАДРАТИ-
  ЧНАЯ АППРОКСИМАЦИЯ
8-ЧЕБЫШЕВСКАЯ АППРОКСИМАЦИЯ
```

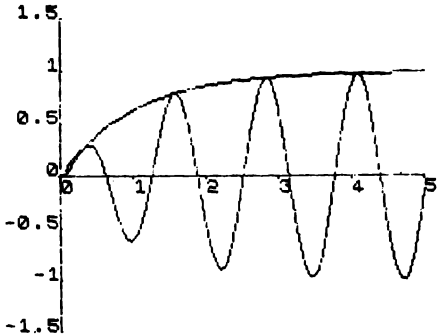
Это построение графиков произвольных функций, заданных аналитически (с вводом по указанию ПЭВМ) и подпрограммой, аппроксимированных описанными выше методами (полиномом заданного порядка и глобальным сплайном). Кроме того, дополнительно предусмотрен вывод графиков функций, аппроксимированных локальными сплайнами, отрезками парабол (многоинтервальная квадратичная аппроксимация), полиномом, полученным методом наименьших квадратов и полиномом Чебышева (описание этих методов дано в [1]). Предусмотрено наложение кривых друг на друга при одних и тех же исходных данных или различных, а также нанесение на кривые узловых точек. Построение координатных осей и их оцифровка происходят автоматически после запроса минимальных и максимальных значений x и y. В программу заложен детальный диалог с пользователем, практически исключающий ошибочные действия последнего.

Рис. 7.48,*а* иллюстрирует построение графика функции $f(x) = (1 - e^{x})\sin(5x)$, вводимого по запросу ПЭВМ в виде `F(X) = (1 - EXP - X) * * SIN(5*X)`. На него наложен график огибающей $1 - e^{-x}$. При построении были заданы $x_{min} = 0$, $x_{max} = 5$, $y_{min} = -1{,}5$ и $y_{max} = 1{,}5$.

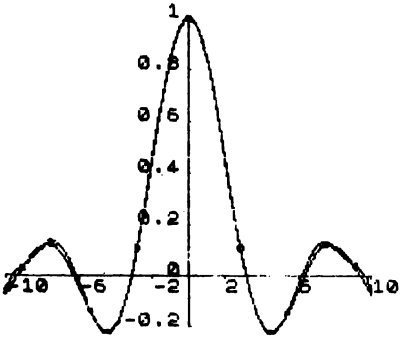
На рис. 7.48,*б* показано построение графика функции $f(x) = (\sin x)/x$, заданного подпрограммой (строки 3200–3299) при $x_{min} = -10$, $x_{max} = 10$, $y_{min} = -0{,}2$ и $y_{max} = 1$. На график наложен другой – тоже функция, аппроксимированная полиномом Чебышева порядка $M = 10$. Выведены и узловые точки.

Действия в остальных режимах иллюстрируем примером – аппроксимацией входной вольт-амперной характеристики ТТЛ-элемента, заданной таблицей:

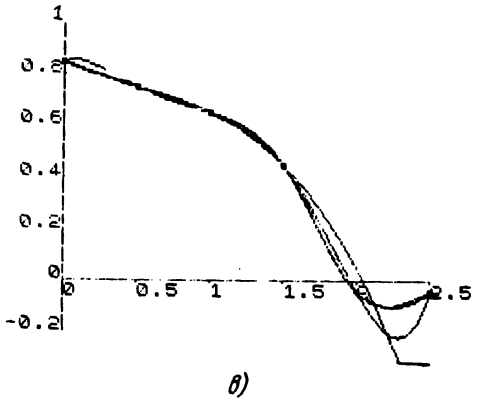
i	0	1	2	3	4	5
$x = U$, В	0	0,5	1	1,5	2	2,5
$y = I$, мА	0,85	0,75	0,65	0,45	–0,04	– 0,04



а)



б)



в)

Рис. 7.48

На рис. 7.48, *в* дан результат наложения графиков этой характеристики: аппроксимация обычным полиномом (степень 5), полиномом степени 4 по методу наименьших квадратов (МНК), глобальным и локальным кубическими сплайнами и многоинтервальная квадратичная аппроксимация. Кривые для трех последних аппроксимаций практически сливаются и, несмотря на малое число узлов (6), хорошо описывают данную функцию. В то же время полиномиальная аппроксимация (особенно МНК) в данном случае неудовлетворительно описывает функцию после резкого ее изгиба.

Таким образом, программа UGD является мощным инструментом по анализу и математическому представлению различных нелинейных зависимостей. Во всех режимах ее работы наряду с выводом графиков предусмотрена выдача числовой информации: таблицы x и y в узловых точках и коэффициентов полиномов и сплайнов. Имеется также возможность вычисления $f(x)$ при любых x с интерполяцией всеми отмеченными выше способами. Комментарии REM позволяют разобраться в назначении и действии отдельных частей этой программы.

Программа R16 (см. программу П1.7) обеспечивает проведение линейной и нелинейной регрессии для 16 зависимостей $y(x)$, заданных точками x_i и y_i.

В строках 1–16 – вывод меню регрессий:

```
РЕГРЕССИЯ ДЛЯ 16 Y(X)

МЕНЮ ФУНКЦИЙ:
1 Y=A+B*X
2 Y=1/(A+B*X)
3 Y=A+B/X
4 Y=X/(A+B*X)
5 Y=A*B↑X
6 Y=A*EXP(B*X)
7 Y=A*10↑(B*X)
8 Y=1/(A+B*EXP(-X))
9 Y=A*X↑B
10 Y=A+B*LOG(X)
11 Y=A+B*LN(X)
12 Y=A/(B+X)
13 Y=A*X/(B+X)
14 Y=A*EXP(B/X)
15 Y=A*10↑(B/X)
16 Y=A+B*(X↑N)
```

15–105 – ввод данных и вывод результатов в числовой форме, 110–400 – построение координатных осей и их автоматическая оцифровка, 410–490 – построение графиков $y(x)$, 1000–1030 – линейная регрессия. Нелинейные зависимости $y(x)$ преобразуются в линейные с помощью линеаризации при вводе x_i и y_i (строки 1200–1360) и вычислении параметров A и B аппроксимаций $f(x)$ (строки 2000–2160).

Подробно эти преобразования описаны в [1], но отчетливо видны и непосредственно из указанных фрагментов программы R16. Предусмотрена выдача результатов – параметров A, B и коэффициента взаимной корреляции R – в цифровой и графической форме, нанесение узловых точек $y_i(x_i)$ наложение кривых друг на друга при одинаковых или различных исходных данных и вычисление $y(x)$ при заданном x по любой из 16 зависимостей (строки 2400–2560).

На рис. 7.49 показан график (построенный по узловым точкам) зависимости $y = Ax^B$ (позиция 9 меню) при вводе следующих данных:

i	1	2	3	4	5	6
x_i	1	2	3	4	5	6
y_i	3	12	27	48	75	108

Конец ввода x_i и y_i задается вводом N(N∅). В этом случае кривая $y(x)$ хорошо описывает исходную совокупность узловых точек $y_i(x_i)$. Для сравнения на эту кривую наложена кривая функции $y(x) = Ae^{Bx}$ (позиция 6 меню). Для нее погрешность аппроксимации значительна. Отметим, что чем ближе к 1 коэффициент взаимной корреляции R, тем точнее $y(x)$ описывает совокупность точек $y_i(x_i)$. В отличие от сплайн-аппроксимации и обычной полиномиальной, для которых кривая $y(x)$ проходит точно через узловые точки зависимости $y(x)$, полученные при регрессии минимизируют среднеквадратическое отклонение $y(x)$ от $y_i(x_i)$. На них меньше сказывается случайная погрешность задания $y_i(x_i)$. Выбор необходимой зависимости $y(x)$ из меню может быть формальным – выбирается та, для которой R ближе к 1. Однако часто на вид зависимости указывают физические или математические закономерности. Например, вольт-амперная характеристика

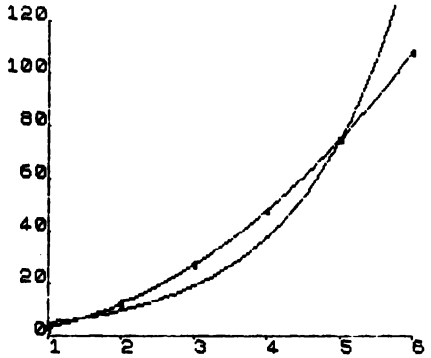


Рис. 7.49

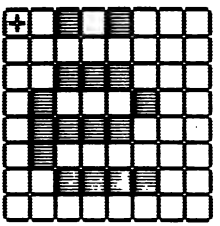


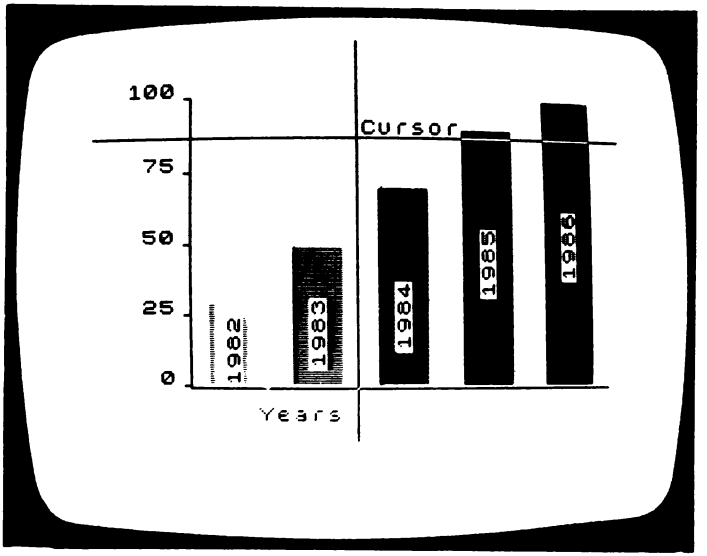
Рис. 7.50

полупроводникового диода – экспоненциальная зависимость. Известны случаи, когда перебор $y(x)$ позволил выявить неизвестные ранее математические закономерности явлений, лежащих в основе зависимости $y_i(x_i)$, и затем обосновать их.

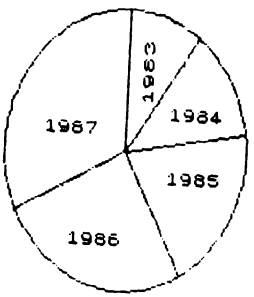
В [52] дается описание бейсик-программы CHARACTER GENERATOR (с ее листингом). Программа позволяет создавать около 400 различных знаков в матрице знакоместа 8×8, а также наблюдать все знаки одновременно, знаки отдельных наборов, копировать наборы, редактировать знаки, записывать их на магнитный носитель и загружать с него в ОЗУ ПЭВМ. Процесс редактирования с помощью этой программы – латинская буква "е" превращена в русскую "ё" с добавлением двух точек сверху показан на рис. 7.50. Имеется возможность вращения (ROTATION) знаков, поворота их вокруг оси X или Y и дублирования четырех четвертей в виде четырех графем (для создания знаков удвоенного размера).

В [52] приведена также универсальная программа графики DIAGRAM AND DATA GRAPS (сокращенно DDG), которая обеспечивает: загрузку спецзнаков (в том числе для нанесения надписей вдоль оси Y) и рисунков с магнитного носителя, построение линейных и круговых гистограмм с различной штриховкой и раскраской, нанесение на графики поясняющих надписей, вывод курсора с сетки знакомест, построение точек, линий и окружностей. Диалог с программой понятен даже малоопытному пользователю.

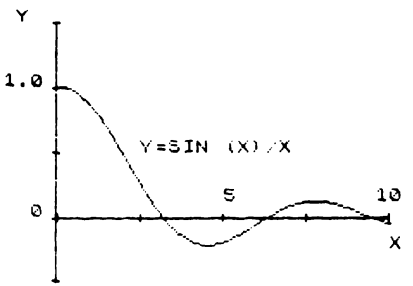
Построенные с помощью программы DDG линейная гистограмма, круговые гистограммы без штриховки и сдвига сегментов и со штриховкой и сдвигом сегментов показаны на рис. 7.51,*а–в*. Графики функций программа DDG строит без оцифровки осей. На рис. 7.51,*г* показан график функции $y = \sin x/x$ при $x_{min} = 0{,}001$, $x_{max} = 10$, $y_{min} = -0{,}5$ и $y_{max} = 1{,}5$. В данном случае x_{min} нельзя брать равным 0: при $x = 0$ ПЭВМ остановит счет и выдаст сигнал ошибки (деление на 0), хотя в действительности $(\sin x)/x > 1$ при $x = 0$. Малое, но конечное x_{min} на графике практически незаметно, но исключает такую ситуацию.



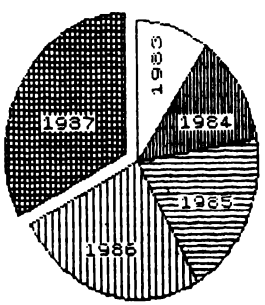
а)



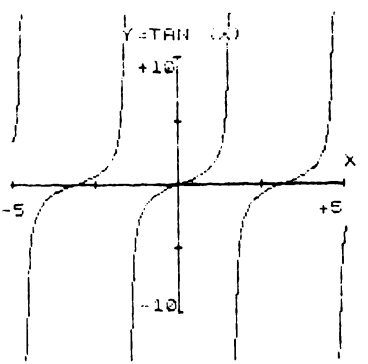
б)



г)



в)



д)

Рис. 7.51

С помощью этой программы нельзя строить и графики функций, кривые которых выходят за пределы окна экрана (при этом также индицируется ошибка и счет останавливается). Однако с помощью условных переходов, например

```
IF  Y<=Ø THEN LET Y=Ø
IF  Y<=175 THEN LET Y=175
```

можно ограничить пределы изменения (эти команды надо ввести в программу перед ее графической частью). При этом подобные функции можно строить одновременно с графиком функции $y = \mathrm{tg}\, x$, периодически уходящей в бесконечность (рис. 7.51, *д*).

Еще большие возможности имеют программы, использующие подпрограммы в машинных кодах. Одна из таких программ ARTIST позволяет создавать цветные картинки: художественные орнаменты, фигуры для вышивки, мозаики, спрайты большого размера, комплекты картин для мультипликационных машинных фильмов и т.д. Эффективное применение таких программ требует профессиональных навыков работы с ними, хотя диалог, заложенный в их основу, достаточно прост и напоминает описанный выше. Программы этого типа (нередко занимающие всю емкость ОЗУ) относятся к категории программ для постоянного использования в специальных целях, например при художественных работах. Подобные программы есть в библиотеке прикладных программ любой массовой ПЭВМ.

7.12. АНАЛИЗ И МОДЕЛИРОВАНИЕ ЭЛЕКТРОННЫХ СХЕМ

Для расчета электронных схем (например, линейных) используют обычно аналитические методы. Однако нередко они столь сложны, что требуют применения ЭВМ. Так, для расчета даже резисторных цепей методами узловых потенциалов или контурных токов требуется составить и решить системы из N линейных уравнений. Решение таких систем при $N>4-5$ вручную или с помощью ПМК довольно трудоемко и чревато ошибками. Расчет нелинейных схем, как правило, выполняется на ЭВМ или графическими методами.

Под анализом электронных схем на ЭВМ (машинный анализ) обычно подразумевается анализ, в ходе которого по заданной топологии цепи автоматически составляется и решается система уравнений (алгебраических или дифференциальных), описывающих работу схем [39, 42, 43]. Обычно такой анализ довольно сложен и реализуется с помощью больших ЭВМ. Однако в последнее время с ними успешно конкурируют и персональные ЭВМ.

Анализ схем с помощью дискретных моделей. Машинный анализ часто выполняется на основе применения дискретных моделей элементов, в общем случае нелинейных. Однако для малых изменений напряжений и токов, что характерно для одного шага h расчета переходных процессов, элементы можно считать линейными, т.е. принять, что их параметры постоянны в пределах шага. При этом они изменяются от шага к шагу, это позволя-

ет учесть нелинейность или параметрическое (во времени) изменение элементов.

Подобный подход успешно используется и для построения моделей реактивных элементов C и L. Более того, в этом случае удается свести их к моделям, содержащим только источники ЭДС и тока с резисторами. Например, ток, протекающий через конденсатор C, $i_C = C(du/dt)$. В пределах малого шага $\Delta t = h$ вместо du/dt можно записать

$$\frac{du}{dt} \approx \frac{u_{n+1} - u_n}{h},$$

где u_n, u_{n+1} – напряжения на конденсаторе C на предшествующем данном шаге. Из этих формул вытекает, что

$$i_{n+1} \approx (C/h)\, u_{n+1} - (C/h)\, u_n. \qquad (7.5)$$

Следовательно, на $(n+1)$-м шаге емкость C можно приближенно представить в виде проводимости $G = C/h$, через которую течет ток $(C/h)u_{n+1}$, и источника тока $I = (C/h)\,u_n$, включенных параллельно. Если $C = \text{const}$, то ток I зависит от напряжения u_n в конце предшествующего шага. Однако если $C \neq \text{const}$, то G и I зависят от напряжения u_n и u_{n+1}.

Аналогично (7.5) для индуктивного элемента имеем

$$u_{n+1} \approx (L/h)\, i_{n+1} - (L/h)\, i_n.$$

Следовательно, индуктивность L представляется в виде последовательно включенных резистора с сопротивлением $R = L/h$ и источника напряжения $(L/h)\,i_n$. Преобразовав источник напряжения в источник тока, можно заменить индуктивный элемент L параллельно включенными проводимостью $G = h/L$ и источником тока i_n (т. е. тока, текущего через L на предшествующем шаге).

Описанное преобразование L и C позволяет на каждом шаге h анализа составить по законам Кирхгофа линейную схему замещения цепи, содержащую только источники ЭДС или тока и реактивные элементы (это составление возлагается на ЭВМ). Используя, например, метод узловых напряжений, определение последних можно свести к решению системы линейных уравнений вида

$$\mathbf{GU} = \mathbf{I} \quad \text{или} \quad \mathbf{U} = \mathbf{IG}^{-1}.$$

где $\mathbf{U}$ – вектор узловых напряжений (неизвестных); $\mathbf{I}$ – вектор совокупных действующих токов; $\mathbf{G}^{-1}$ – обращенная (инвертированная) матрица проводимостей. Если R, L и C линейны, то матрица проводимостей $\mathbf{G}$ не меняется от шага к шагу. Следовательно, ее достаточно обратить один раз. Тогда на каждом шаге можно найти значения U, умножив сформированный вектор действующих токов (включая входящие в дискретные модели) на обращенную матрицу проводимостей. Если R, L и C нелинейные, матрицу $\mathbf{G}$ приходится формировать и обращать на каждом шаге вычислений, что увеличивает время вычислений. В этом случае лучше искать вектор $\mathbf{U}$, решая на каждом шаге систему уравнений, например, методом Гаусса.

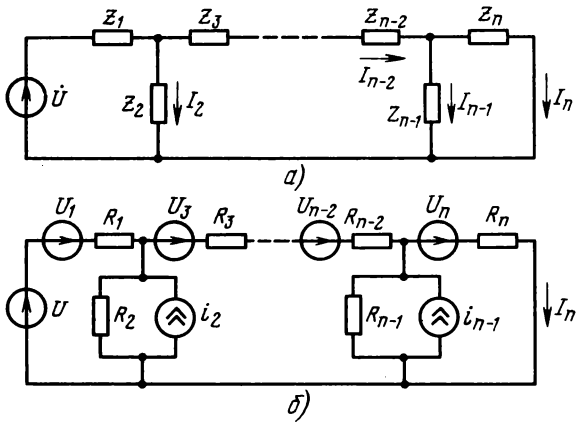


Рис. 7.52

Анализ лестничных цепей. Некоторые специальные виды цепей можно анализировать более простыми методами. Примером служат лестничные цепи, в которых к одному источнику ЭДС или тока подключена цепочка пассивных элементов. К таким цепям относятся многие виды резонансных цепей, фильтров, фрагменты схем усилительных каскадов и т. д.

Алгоритм расчета лестничных цепей следующий [39]. Допустим цепь представлена в виде рис. 7.52,*а*. Пусть U_k – напряжение k-й ветки, а I_k – суммарный ток (протекающий через резистор R_k, включенный параллельно с источником тока i_k). Вычисления начинаем с последней ветви, для которой $I_n = I_{n-1} = U_n/R_n + i_n$. Далее можно записать

$$U_{n-1} = (I_{n-1} - i_{n-1})/R_{n-1} = (R_{n-1}/R_n)\,U_n + (i_n - i_{n-1})R_{n-1},$$
$$U_{n-2} = U_{n-1} + U_n = (1 + R_{n-1}/R_n)\,U_n + (i_n - i_{n-1})R_{n-1}.$$

Итак, в общем случае

$$U_k = a_k U_n + b_k;\ I_k = c_k U_n + d_k. \qquad (7.6)$$

Коэффициенты a_k, b_k, c_k и d_k могут вычисляться последовательно, начиная с последней ветви:

$$a_n = 1, b_n = 0, c_n = 1/R_n, d_n = i_n;$$
$$a_{n-1} = R_{n-1}/R_n, b_n = (i_n - i_{n-1})R_{n-1}, c_{n-1} = 1/R_n, d_{n-1} = i_n;$$
$$a_{n-2} = 1 + R_{n-1}/R_n, b_{n-2} = (i_n - i_{n-1})R_{n-1}, \ldots$$

Выполнив численный расчет всех n коэффициентов, U_n находим из уравнения входного контура

$$u = U_1 + U_2 = (a_1 + a_2)\,U_n + (b_1 + b_2),$$

откуда

$$U_n = [u - (b_1 + b_2)]/(a_1 + a_2).$$

Затем из (7.6) вычисляются U_k и I_k других ветвей.

Приведенная в приложении программа расчета линейных лестничных цепей (программа П1.8) позволяет рассчитать переходные процессы в цепи, содержащей до 50 ветвей. В каждой ветви могут включаться по три различных элемента ($G = 1/R$, L и C), в общей сложности до 150 пассивных элементов. Нумерация ветвей должна начинаться с 1 и с последней ветви, а заканчиваться номером n = N, первой, подключенной к источнику возбуждения ветви (ветвь 1 должна быть вертикальной, что всегда выполнимо). В последовательных ветвях элементы L, C и R могут включаться последовательно, но в точку их соединения включается пронумерованная ветвь с резистивной нулевой проводимостью ($G = 0$).

1. Задание топологии лестничной цепи и ввод данных. Данные о топологии схемы задаются указанием числа ветвей N, типа элемента (G, L, C) и номера ветви. Ввод этих данных сводится к составлению протокола ввода–таблицы с заголовком

ТИП ЭЛЕМЕНТА НОМЕР СТРОКИ НОМИНАЛ

Перед вводом каждой строки таблицы ПЭВМ задает пользователю вопрос:

УКАЖИТЕ ТИП (G, L, C, F, K, Ø)?

Здесь G – проводимость, L – индуктивность, C – емкость, F – воздействие, K, Ø – указание о коррекции и о конце ввода. При вводе G, L и C после указания типа элемента вводится номер ветви и номинальное значение элемента. Это ведет к заполнению соответствующей строки таблицы – протокола ввода.

2. Задание воздействия. Воздействием может быть напряжение или ток. Если число ветвей четно, то воздействие задается источником напряжения, если иначе, то источником тока. ПЭВМ определяет это автоматически и присваивает амплитуде воздействия заданное значение с соответствующей единицей измерения – вольты (В) или амперы (А). При задании типа воздействия F ПЭВМ дает указание

ВВЕДИТЕ A (A или B), W (УГЛ.), QØ (РАД), ТИ (С)

Здесь A – амплитуда; W – угловая частота; QØ – начальная фаза; ТИ – длительность воздействия.

Если задать все эти параметры, то воздействие будет иметь вид косинусоиды:

$x(t) = A\cos(Wt + Q_0)$ при $0 < t \leqslant t_и$,
$x(t) = 0$ при $t > t_и$.

При задании $W = 0$ и $Q_0 = 0$ воздействие имеет вид прямоугольного импульса

$x(t) = A$ при $0 < t \leqslant t_и$,
$x(t) = 0$ при $t > t_и$.

В частности, при $t_и$ = ТИ = ТК, где ТК – задаваемое в дальнейшем конечное время ТИ, воздействие имеет вид одиночного перепада.

При задании $W = 0$, $Q_0 = 0$ и $t_и < 0$ воздействие имеет вид экспоненциального перепада $x(t) = A(1 - e^{-t/t_и})$.

3. Коррекция ввода. Если по ходу заполнения таблицы данных выявляется ошибка (что не редкость при задании сложных схем), то вводится К (указание о коррекции). В ответ ПЭВМ дает указание:

УКАЖИТЕ ТИП КОРРЕКТИРУЕМОГО ЭЛЕМЕНТА

После этого надо повторно ввести скорректированные данные о корректируемом элементе.

4. Пуск вычислений. После ввода данных о цепи надо указать тип ввода К. При этом ПЭВМ запрашивает значения шага h = H и ТК. Введя их, пользователь получает сообщение:

ВВОД ЗАКОНЧЕН
НАЖМИТЕ ЛЮБУЮ КЛАВИШУ

Нажатие любой клавиши ведет к пуску вычислений, по мере которых на экран дисплея выводится значение текущего времени t = T (с шагом h = H) и таблица напряжений и токов каждой ветви для указанного значения T. Закончив вычисления, ПЭВМ дает указание

КОНЕЦ ВЫЧИСЛЕНИЙ
ЕСЛИ НУЖЕН ВЫВОД ГРАФИКОВ
НАЖМИТЕ ЛЮБУЮ КЛАВИШУ

Нажав любую клавишу, можно ввести в действие часть программы для построения заданных в ней графиков (всех токов и напряжений ветвей или тех, которые интересуют пользователя).

Пример. Пусть нужно проанализировать переходные процессы при заряде от источника ЭДС (единичный перепад E=1 В) накопительной линии с дискретными элементами L =1 Гн и C=1 Ф; заряд происходит через резистор R =1 Ом (рис. 7.53). Копии изображений иллюстрируют диалог с ПЭВМ – первые две кратко напоминают о работе с программой, третья – протокол ввода:

```
    АНАЛИЗ ЛЕСТНИЧНЫХ ЦЕПЕЙ
  МЕТОДОМ ДИСКРЕТНЫХ МОДЕЛЕЙ

ВНИМАНИЕ!  НУМЕРАЦИЯ ВЕТВЕЙ НАЧИ
НАЕТСЯ С ПОСЛЕДНЕЙ ВЕТВИ.ДОПУСКА
ЕТСЯ ПАРАЛЛЕЛЬНОЕ СОЕДИНЕНИЕ R,C
,L В ЛЮБОЙ ВЕТВИ.ПРИ ПОСЛЕДОВАТЕ
ЛЬНОМ СОЕДИНЕНИИ В ГОРИЗОНТАЛЬНО
Й ВЕТВИ К ТОЧКЕ СОЕДИНЕНИЯ ВКЛЮЧ
АЕТСЯ ПРОНУМЕРОВАННАЯ ВЕРТИКАЛЬН
АЯ ВЕТВЬ С GI=0
 ЗАПИСЬ ТОПОЛОГИЧЕСКИХ МАССИВОВ:
G,I,NOM-МАССИВ ПРОВОДИМОСТЕЙ
C,I,NOM-МАССИВ ЕМКОСТЕЙ
L,I,NOM-МАССИВ ИНДУКТИВНОСТЕЙ
F,A,ω0,ТИ-МАССИВ ВОЗДЕЙСТВИЙ
 ГДЕ I-НОМЕР ВЕТВИ,A-АМПЛИТУДА,ω
-ЧАСТОТА,ω0-НАЧАЛЬНАЯ ФАЗА,ТИ-ДЛ
ИТЕЛЬНОСТЬ ВОЗДЕЙСТВИЯ

 НАЖМИТЕ ЛЮБУЮ КЛАВИШУ


     ВИДЫ ВОЗДЕЙСТВИЙ:

1.ω=0,ω0=0,ТИ<ТК - ПРЯМОУГОЛЬНЫЙ
 ИМПУЛЬС С ДЛИТЕЛЬНОСТЬЮ ТИ

2.ω=0,ω0=0,ТИ=ТК - ПЕРЕПАД
```

```
3,W=0,00=0,ТИ<0 - ЭКСПОНЕНЦИАЛЬН
ЫЙ ПЕРЕПАД F=A*(1-EXP(t/ТИ))

4,W<>0,00,ТИ - ГАРМОНИЧЕСКОЕ ВОЗ
ДЕЙСТВИЕ F=A*COS(W*t+00) ПРИ 0<t
<=ТИ И F=0 ПРИ ТИ<t<ТК

НАЖМИТЕ ЛЮБУЮ КЛАВИШУ


НАЧИНАЙТЕ ВВОД:
ЧИСЛО ВЕТВЕЙ N=8
ТИП   ВЕТВЬ   НОМИНАЛ     ЕД.ИЗМЕР.
C     1       1           Ф
L     2       1           ГН
C     3       1           Ф
L     4       1           ГН
C     5       1           Ф
L     6       1           ГН
G     7       0           1/ОМ
G     8       1           1/ОМ

 ПАРАМЕТРЫ ИСТОЧНИКА:
A=1 В              W=0 РАД/С
00=0 РАД           ТИ=10 С
H=0.1
TK=10
ВВОД ЗАКОНЧЕН
НАЖМИТЕ ЛЮБУЮ КЛАВИШУ
```

Стоп-кадр вывода T, U, J = *I* для 8 ветвей цепи на рис. 7.53 и графики переходных процессов – входное напряжение на ветвях 1, 3, 4, 5 и 8 – приведены на рис. 7.54. Из временно́й зависимости U8 видно, что напряжение на резисторе ветви 8, а следовательно, и ток заряда линии имеют форму импульса с довольно плоской и протяженной вершиной. При большем числе ветвей форма импульсов приближается к прямоугольной.

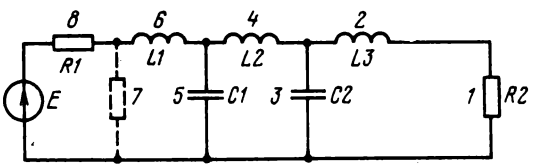


Рис. 7.53

```
РЕЗУЛЬТАТЫ ВЫЧИСЛЕНИЙ:
T=0.1              F(T)=1
U1=8.6607756E-7    J1=8.6607756E-6
U2=.000086607756
J2=8.6607756E-6
U3=.000087473834
J3=.000087473834
U4=.0088339911     J4=.00088339911
U5=.089921465      J5=.08921465
U6=0.90098049      J6=.090098049
U7=0.909980195     J7=0
U8=.090098049      J8=.090098049
T=0.2              F(T)=1
U1=5.90180655E-6   J1=.000005035529
U2=.00041694514    J2=.00005035529
U3=.0042294675     J3=.0033537291
U4=.025206653      J4=.0034040844
U5=.2256297        J5=0.18708235
U6=0.80388387      J6=0.17048643
U7=0.82951356      J7=0
U8=0.17048643      J8=0.17048643
```

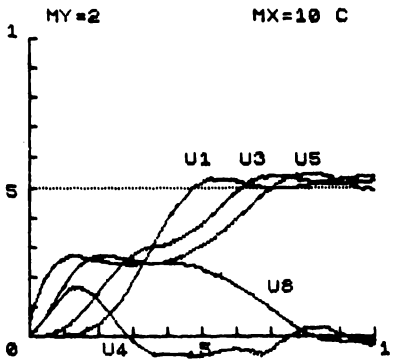


Рис. 7.54

Программа содержит следующие основные блоки:

Номера строк	Основные блоки
10–60	Начальный диалог, комментарий
70–152	Ввод данных (G, L, C, F, K, ∅)
154–164	Коррекция ввода данных
165–230	Ввод H и TK: подготовка к вычислениям
240–720	Расчетная часть программы
730–1130	Построение графиков
9600–9635	Построение координатных осей

Расчетная часть программы соответствует описанному выше алгоритму анализа лестничных цепей. Программа построения графиков должна несколько подстраиваться под задачи пользователя. Как правило, нецелесообразно выводить все графики. Например, для цепи на рис. 7.54 это потребует вывода 17 графиков, и разобраться, чему соответствует каждый, просто невозможно. В нашем случае задан вывод графиков только напряжений ветвей, да и то не всех (в строке 1055 задано исключение из выдачи графиков U2, U6 и U7). Под выводимые графики подстраиваются и поясняющие надписи (строки 1100–1120, 9630, 9635), а также масштабы по осям X и Y и подпрограмма построения координатных осей. Полезно иметь несколько подпрограмм вывода координатных осей и вводить их отдельно, соединяя с помощью оператора MERGE с основной программой.

Анализ линейных и нелинейных схем произвольной конфигурации. Для машинного анализа таких схем наиболее удобен метод узловых потенциалов с применением дискретных моделей для элементов R, L, C и I.

При линейных R, L и C, но зависимых токах I матрица узловых проводимостей обращается один раз, а при нелинейном хотя бы одном элементе R, C и L ее приходится обращать на каждом шаге вычислений. Соответственно программа анализа схем произвольной конфигурации должна иметь два отмеченных режима работы.

Источники напряжения должны быть преобразованы в источники тока. Расчет переходных процессов начинается при нулевых начальных условиях. Шаг h = H должен быть меньше половины наименьшей из постоянных времени схемы.

1. Задание топологии схемы и ввод данных. Работа программы П1.8 начинается с выдачи двух страниц с инструкциями и запроса:

ИМЯ РАСЧЕТА ?

Введенное по запросу наименование расчета (или анализируемой схемы) выводится на экран дисплея в виде заголовка протокола ввода. После этого следует запрос:

ЧИСЛО УЗЛОВ = ?

Необходимо ввести общее (включая общий узел) число узлов. Затем следует выполнить указание

ЗАДАЙТЕ L ЕСЛИ R, L, C ЛИНЕЙНЫ И
N ЕСЛИ ОНИ НЕЛИНЕЙНЫ

Характер линейности R, L и C заносится в протокол ввода.

Анализируемая схема должна быть задана эквивалентной схемой с пронумерованными узлами. Последним нумеруется общий узел. Нумерация начинается с 1. В схеме могут быть элементы R, C, L и I. Линейные элементы задаются непосредственно. Нелинейные, зависимые и параметрические элементы должны быть заданы моделями и подпрограммами. Каждый элемент характеризуется типом (R, C, L, I), номерами Y1 и Y2 узлов, к которым он подключен, и двумя характеристическими параметрами F и NOM. Должен быть составлен протокол ввода в виде таблицы с заголовком

ТИП Y1 Y2 F NOM

Для элементов R, C, L параметр F задает номер начальной строки НС подпрограммы, который вводится со знаком минус. Значение $F \geqslant \emptyset$ может использоваться как особый параметр элемента, обычно источника тока. Параметр NOM указывает номинальное значение (номинал) элемента. Если R, L или C вычисляется по подпрограмме, параметр NOM задает номер узла, от напряжения которого они зависят.

Примеры ввода линейных элементов R, L, C:

R1 1 2 $\emptyset$ 5.1

– вводится сопротивление резистора R1, включенного между узлами 1 и 2, с номиналом 5.1 Ом (значение $F = \emptyset$ может быть любым, в частности 0)

C2 3 4 0 1E–9

– вводится емкость конденсатора C2, включенного между узлами 3 и 4 с номиналом $1 \cdot 10^{-9}$ Ф. Номера у однотипных элементов (например, C1, C2, C3 и т.д.) проставляются ПЭВМ автоматически. Ввод выполняется по запросу

ТИП (R, C, L, I, K, $\emptyset$)?

2. Задание встроенных воздействий. Воздействие в виде источника тока задается типом ввода I, параметрами F и NOM, а также временными параметрами ТЗ (время задержки), TI (длительность воздействия), TF (время нарастания и спада или постоянная времени их) и ТК (конечное время счета).

Параметр $F > -2000$ задает значение пьедестала тока, а NOM – амплитуду воздействия или для зависимых источников тока номер управляющего узла. В основной программе воздействие задается в виде расположенного на пьедестале импульса с экспоненциальным нарастанием и спадом:

$I1 = 0$ при $T < T3$,

$I1 = I_{ном}(1 - EXP(-(T-T3)/TF)) + I$ при $T3 \leqslant T \leqslant T2$,

$I1 = IM \cdot EXP((T - T2)/TF) + I$ при $T > T2$,

где $IM = I_{ном}(1 - EXP((TI-T3)/TF))$; $T2 = TI + T3$.

Варьируя $I_{ном}$, I_F, TЗ, TI и TF, можно создавать различные виды воздействий. Например, при TI = TF и $I_F = 0$ воздействие имеет вид единичного экспоненциального перепада, при $I_{ном} = 0$ и $I_F \neq 0$ – скачка тока (постоянный ток при $t = t \geqslant 0$), при TF $\ll$ H воздействие получается в виде прямоугольного импульса или одиночного ступенчатого перепада. Практически любые виды воздействий задаются с помощью подпрограмм.

3. Библиотека подпрограмм и моделей. Число нелинейных элементов столь велико, что невозможно включить все модели их в программу. Это и нерационально: большинство из них будет просто не использоваться, бесполезно занимая ОЗУ и снижая степень сложности анализируемых схем.

В данной программе предусмотрено включение в нее практически любых моделей в виде подпрограмм. Эти подпрограммы образуют библиотеку моделей. Нет необходимости хранить ее в ОЗУ. Более удобно записать подпрограммы на магнитные носители (диски или кассеты) и с помощью оператора MERGE загружать в ПЭВМ только нужные для анализа подпрограммы (модели).

Модели могут содержать описания нелинейных (зависимых) элементов R, L и C, а также зависимых источников тока. Номер строки, с которой начинается подпрограмма, задается как F = –HC, причем на подпрограммы выделена область строк, начиная со строки 2000. Параметр NOM может задавать номинал, масштабный множитель или номер управляющего узла. В программе числовое значение этого параметра присваивается специальной переменной Y. Напряжения узлов задаются массивом Y(I), например Y(1) есть напряжение первого узла, Y (Y) – напряжение узла, номер которого задан переменной Y = NOM.

Перед обращением к каждой подпрограмме программа формирует текущие номера узлов K и K1, к которым подключен данный элемент. Это позволяет в общем виде определить напряжения Y(K) и Y(K1) узлов или разность (Y(K) – Y(K1)) этих напряжений. Напряжения Y(K) и Y(K1) определяются относительно общего узла (его потенциал считается равным 0). Таким образом, нелинейные элементы, параметр которых зависит от разности напряжений между их выводами, могут задаваться в общем виде. Это позволяет пользоваться моделью неоднократно, причем каждый раз имеется возможность изменять ее параметр Y = NOM.

Параметром Y = NOM можно задать также номер произвольного узла. Это полезно при анализе сложных схем. Например, ток стока полевого транзистора, включенного в сложную схему, является функцией напряжения на затворе. Если затвор подключить к произвольному (по номеру) узлу P, то, задав NOM = P, мы получим возможность задания управляемого напряжением этого узла источника тока.

Разумеется, можно записывать подпрограммы, указывая в них непосредственно требуемые номера узлов. Вычисляемые по подпрограммам значения R, L и C должны присваиваться соответственно переменным R, L и C, а значения тока I должны присваиваться переменной I1.

Для иллюстрации техники составления подпрограммы в программе расчета электронных цепей с применением дискретных моделей (программа П1.9) включен ряд подпрограмм (со строки 2000).

Дополнительные воздействия. Заданы подпрограммы формирования импульса длительностью TI трапецеидальной формы с линейным (за время TF) нарастанием и спадом (строки 2010–2040) и подпрограмма синусоидального воздействия I1 = = Y + Y * SIN(2 * PI * T/TI + TF), имеющего период TI и временной сдвиг TF (строки 2100 и 2110). Эти две подпрограммы иллюстрируют задание дополнительных воздействий с различными временными зависимостями тока I1. В этих воздействиях время задержки TЗ = 0.

Модель туннельного диода. В строках 3000 и 3010 задан зависимый источник тока $I_1 = A u e^{-\alpha u} + D(e^{\beta u} - 1)$, описывающий статическую модель туннельного диода при $A = e^Y/U_1$, $U_1 = 0{,}1$ В (напряжение пика ВАХ), u=U2, $D = 1 \cdot 10^{-8}$ А, $\alpha = 10$ В$^{-1}$ и $\beta = 20$ В$^{-1}$. Эти данные типичны для туннельных GaAs-диодов. В данном случае задана зависимость I_1 от напряжения второго узла, а параметр Y = NOM используется как масштабный множитель, определяет ток пика $I_П$ N-образной ВАХ туннельного диода.

В строках 4000 и 4010 записана подпрограмма вычисления нелинейной (зависимой) емкости p–n-перехода

$$C(u) = C_{01}\sqrt{\frac{\varphi_К}{\varphi_К - u}} + C_{02},$$

где $C_{01} = 10 \cdot 10^{-12}$ Ф; $C_{02} = 5 \cdot 10^{-12}$ Ф; $\varphi_К = 1{,}2$ В – контактная разность потенциалов; u = Y(Y). В данном случае управляющее напряжение Y(Y) задается номером Y управляющего узла.

Модель полупроводникового диода. Диод можно представить в виде параллельно включенных зависимого источника тока

$$I = I_0 (e^{\theta U} - 1)$$

и нелинейной емкости

$$C \approx C_0 \sqrt{\frac{\varphi_К}{\varphi_К - U}} + I\theta\tau_Д.$$

Для Si-диода обычно $\theta \approx 20$ В$^{-1}$, а $\varphi_К \approx 1$ В. Поэтому в строках 5000 и 5010 ВАХ диода задана уравнением

I1 = Y * (EXP((Y(K) – Y(K1)) * 20) – 1),

где Y(K) – Y(K1) = U – разность напряжений между узлами, к которым подключены выводы диода. При обращении к статической модели диода (строка 5000 или 5010) пользователь может задавать NOM = Y = I_0, т.е. различные начальные токи I_0 диодов.

Динамическая модель диода дополняется вычислением его барьерной и диффузионной емкости C по формуле

$$C = 5 \cdot 10^{-12}\sqrt{1/(1 - (Y(K) - Y(K1))} + I1 \cdot \theta \cdot Y.$$

Таким образом, здесь задана $C_0 = 5 \cdot 10^{-12}$ Ф, а пользователь может в протоколе ввода (обращения к подпрограмме с начальной строкой 5020) задавать любые постоянные времени диода $\tau_Д$. Смена $\varphi_К$, θ и C_0 достигается изменением их значений прямо в программе. Разумеется, пользователь может модифицировать программу так, что нужные ему параметры моделей будут задаваться в ходе ввода исходных данных. Поскольку это усложняет ввод, такое задание параметров моделей (нередко весьма многочисленных) не всегда желательно.

Модель биполярного транзистора (строки 6000 – 6060) является несколько упрощенной модификацией известной модели Логана, содержит два встречно включенных источника тока

$$I_1 = I_N(1 + 1/\beta_N) - I_I, I_2 = I_1\,(1 + 1/\beta_I) - I_N,$$

где $I_N = I_{SN}(e^{\theta U_Э} - 1)$; $I_I = I_{SI}(e^{\theta U_К} - 1)$. Источник тока I_1 эквивалентен эмиттерному переходу, а источник I_2 – коллекторному.

В процессе подготовки протокола ввода нужно указывать номера узлов, к которым подключены переходы транзисторов, и начальных строк подпрограмм (6000 для эмиттерного перехода и 6020 для коллекторного), а также значения NOM = β_N и NOM = β_I раздельно для переходов. Значения θ, I_{SN} и I_{SI} в программе применять равными θ = = 30 В$^{-1}$, $I_{SN} = 0{,}2 \cdot 10^{-9}$ А и $I_{SI} = 1 \cdot 10^{-9}$ А (значения типичны для маломощных Si-транзисторов).

Источники тока шунтируются нелинейными емкостями (барьерными и диффузионными):

$$C_Э = C_{Э0}\sqrt{\frac{\varphi_К}{\varphi_К - U_Э}} + I_N\theta\tau_N,$$

$$C_К = C_{К0}\sqrt{\frac{\varphi_К}{\varphi_К - U_К}} + I_I\theta\tau_I.$$

В программе вычисление $C_Э$ задается в протоколе ввода обращением к подпрограмме с начальной строкой 6030, а $C_К$ – обращением к подпрограмме с начальной строкой 6040. В ходе ввода можно задавать (как NOM.) $C_{Э0}$ и $C_{К0}$, принято $\varphi_К = 1$ В, $\tau_N = 1 \cdot 10^{-9}$ с и $\tau_I = 10 \cdot 10^{-9}$ с. Значения $U_Э$ и $U_К$ определяются автоматически как разность потенциалов узлов Y (К) и Y (К1) эмиттерного и коллекторного переходов.

С помощью этих моделей можно анализировать разнообразные электронные и электрические схемы. Однако эти модели приведены лишь как образец их задания с помощью подпрограмм. Пользователь по своему усмотрению может дополнять библиотеку подпрограмм, включая в нее дополнительные модели и макромодели самых различных элементов. Еще раз отметим, что нет необходимости хранить их с текстом основной программы в ОЗУ. Правильнее формировать большую часть моделей в виде подпрограмм, записанных на магнитные носители (диски или магнитофонные кассеты), загружая в ОЗУ только нужные для анализа данной схемы подпрограммы.

Приведем примеры обращения к подпрограммам в ходе ввода. Так, указанием в протоколе ввода

```
I1  2  3  –2100  .1
```

задан источник тока, включенный между узлами 2 и 3, изменяющегося синусоидально (подпрограмма начинается со строки 2100) с номиналом 0,1 А; указанием

```
C1  3  4  –4000  2
```

задана нелинейная емкость конденсатора, включенного между узлами 3 и 4 (вычисляется по подпрограмме, начинающейся со строки 4000) и управляемого напряжением второго узла.

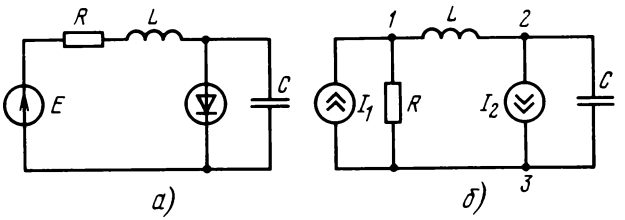


Рис. 7.55

1. Ввод данных. При пуске программы (командой RUN) вначале выводятся две страницы с указанием правил работы с программой, затем запрос об имени расчета и числе узлов N. После указания числа узлов N программа просит ввести символ L, если R, L и C линейны, или N, если они нелинейны. После ввода указания о линейности выводится заголовок таблицы ввода

ТИП Y1 Y2 F NOM.

Правила ввода R, C, L и I были описаны выше. При вводе однотипным элементам автоматически присваиваются порядковые номера. Если значение F = –НС задает обращение к подпрограмме, то в графе F выводится надпись ПП*, после которой идет номер строки НС, начиная с которой должна быть задана подпрограмма.

2. Коррекция ввода. Любой элемент, содержащий ошибку в его характеристиках (значениях Y1, Y2, F или NOM.), можно скорректировать, указав его тип (вместе с порядковым номером). После этого можно полностью повторить ввод, внеся необходимые исправления в данные. Сразу после ввода можно выполнять коррекцию как данных заданного элемента, так и других элементов.

3. Порядок выполнения расчета. После ввода всех R, C, L и I нужно следовать указаниям ПЭВМ (диалог настолько прост, что нет необходимости в его подробном описании). Графическая часть программы позволяет вывести графики одновременно до четырех временных зависимостей: одного из воздействий и напряжений до трех узлов (их номера указываются по запросу ПЭВМ, ∅ означает исключение вывода). Масштаб по оси времен и единицы измерения (с, мс, мкс и нс) указываются автоматически, масштаб по оси напряжений задается по запросу (так как заранее он не всегда известен). По запросу

ЧЕМ СТРОИТЬ (PLOT/DRAW)

можно обеспечить построение графика по точкам (P) или непрерывными линиями (D). Предусмотрена также возможность ввода комментариев в любое место графика, указываемое специальным курсором, который можно перемещать, не нарушая построенные графики. Неудачный комментарий может быть также стерт без нарушения графиков.

Примеры анализа. Пусть нужно построить временные диаграммы работы релаксационного генератора на туннельном диоде (рис. 7.55). Рассмотрим два примера.

Пример 1. Заданы $R_1 = 10$ Ом = const, $L_1 = 0{,}5 \cdot 10^{-7}$ Гн = const, $C_1 = 20 \cdot 10^{-12}$ Ф = = const (т.е. R, L, C линейны), $E = 0{,}3$ В (т.е. $I_1 = E/R_1 = 0{,}03$ А), туннельный диод имеет $I_{II} = 0{,}01$ А. Временные параметры: П = $0{,}1 \cdot 10^{-9}$ с, ТЗ = 0, ТИ = $20 \cdot 10^{-9}$ с, ТФ = $1 \cdot 10^{-9}$, ТК = $20 \cdot 10^{-9}$ с. Ток I_2 задается подпрограммой, записанной со строки 3000. Протокол ввода для данного примера:

```
АВТОКОЛЕБАТЕЛЬНЫЙ РЕЛАКСАТОР
ЧИСЛО УЗЛОВ N=3
R,L,C-ЛИНЕЙНЫ
ТИП   У1 У2 F          НОМ.
R1    1  3  0          10
L1    1  2  0          5E-8
C1    2  3  0          2E-11
I1    3  1  0          .03
I2    2  3  ПП*3000    .01
ШАГ Н=1E-10       ВРЕМЕНА ИМПУЛЬСА
ТЗ=0              ТИ=2E-8
ТФ=1E-9           ТК=2E-8

ВВЕДИТЕ ДАННЫЕ ПОДПРОГРАММ
И ДАЙТЕ КОМАНДУ CONTINUE
```

Вид выдачи данных в числовой форме и графики временных зависимостей для E, U1 и U2 представлены на рис. 7.56. В данном случае релаксационный генератор работает в автоколебательном режиме.

Пример 2. Заданы $R_1 = 10$ Ом = const, $L_1 = 1 \cdot 10^{-7}$ Гн = const, C_1 – нелинейная емкость (вычисляется подпрограммой, начинающейся со строки 4000), I_1 – импульсный ток с амплитудой 0,02 А на пьедестале 0,015 А с экспоненциальным нарастанием и спадом, длительностью ТИ = $5 \cdot 10^{-9}$ с и ТФ = $1 \cdot 10^{-9}$ с, туннельный диод с $I_{II} = 0{,}01$ А (I_2 задается подпрограммой, начинающейся со строки 3000), П = $0{,}1 \cdot 10^{-9}$ с и ТК = = $20 \cdot 10^{-9}$ с. Протокол ввода для этого примера и временные диаграммы E = П * R1, U1 и U2 приведены на рис. 7.57. В данном случае релаксационный генератор работает в ждущем режиме. Отчетливо видна характерная задержка запуска и другие стадии переходных процессов.

В примере 1 время вычисления на одном шаге менее 1 с, а в примере 2 около 3 с (различие связано с тем, что емкость C1 в примере 2 нелинейна). Графики имеют ТК/П = 200 точек.

Персональные ЭВМ с ОЗУ емкостью 48–64 Кбайт позволяют по этой программе анализировать схемы с числом узлов до 50–60 (однако при этом

```
T=1E-10
U1=.027994429
U2=.000277172S7
T=2E-10
U1=.052784726
U2=.00069956158
T=3E-10
U1=.074690428
U2=.0012875374
T=4E-10
U1=.094000119
U2=.0020120906
T=5E-10
U1=0.11097474
U2=.0028851561
T=6E-10
U1=0.12585014
U2=.0038345699
T=7E-10
U1=0.13883949
U2=.00491189
```

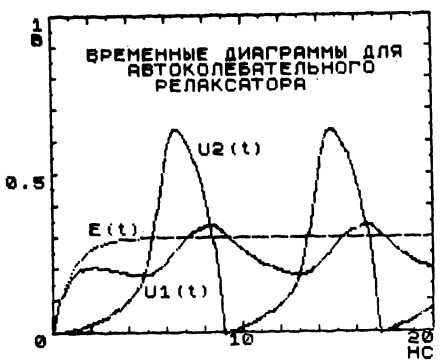


Рис. 7.56

```
ЖДУЩИЙ РЕЛАКСАТОР
ЧИСЛО УЗЛОВ N=3
R,L,C-НЕЛИНЕЙНЫ
ТИП  У1 У2 F          НОМ.
R1    1  3  0          10
L1    1  2  0          1E-7
C1    2  3  ПП*4000     2
I1    3  1  .015        .02
I2    2  3  ПП*3000     .01
ШАГ H=1E-10      ВРЕМЕНА ИМПУЛЬСА
TЗ=1E-9          ТИ=5E-9
T0=5E-10         ТК=3E-8

ВВЕДИТЕ ДАННЫЕ ПОДПРОГРАММ
И ДАЙТЕ КОМАНДУ CONTINUE
```

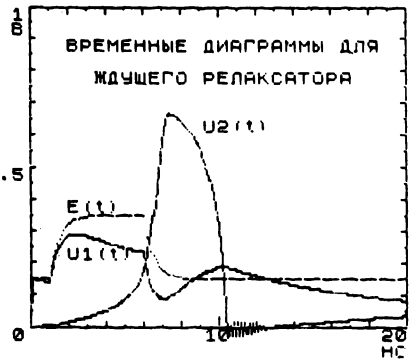


Рис. 7.57

основным ограничивающим фактором становится время вычислений). Число узлов можно увеличить, отказавшись от хранения результатов вычислений в виде массивов и вывода их сразу в виде графиков.

Отметим, что применение программы не ограничивается расчетом переходных процессов. Если анализируется линейная резисторная цепь, то уже на первом шаге программа выдает значения узловых напряжений. Если же цепь нелинейна, но не содержит реактивных элементов, то автоматически реализуется метод Ньютона–Рафсона и после определенного числа итераций (обычно 5–10) вычисляются узловые напряжения для статического режима. Задавая линейное изменение токов от 0 до стационарного значения, можно реализовать метод движения по параметру, значительно улучшающий сходимость метода Ньютона–Рафсона для сложных нелинейных схем.

Определенные трудности при анализе схем с большим разбросом постоянных времени может вызвать выбор шага $h = H$. При H, большем примерно половины наименьшей постоянной времени, может наблюдаться числовая неустойчивость решения вплоть до аварийного останова ПЭВМ из-за появления недопустимо больших чисел. На графиках бывает хорошо заметна и малая периодическая неустойчивость (см. рис. 7.57 кривая U2 при напряжении U2, близком к 0). Неопытный пользователь может отождествлять ее с особенностями переходного процесса в виде быстрых осцилляций, что недопустимо. При подозрении на наличие такой неустойчивости следует провести расчеты при шаге H/2, H/4, ... Если осцилляции остаются, значит, они действительно присущи переходным процессам анализируемой схемы.

Для разбора данной программы можно воспользоваться комментариями в ее тексте, поясняющими назначение отдельных фрагментов программы.

Метод переменных состояния. Из других методов моделирования электронных схем важное значение имеет метод переменных состояния. Под ними подразумеваются величины, не способные меняться мгновенно. Обычно это напряжения на конденсаторах или токи, текущие через индуктивные элементы. Метод переменных состояния применим при любых начальных условиях. Как правило, он реализуется гораздо более простыми программами (для частных схем), чем описанные выше. Практически данный

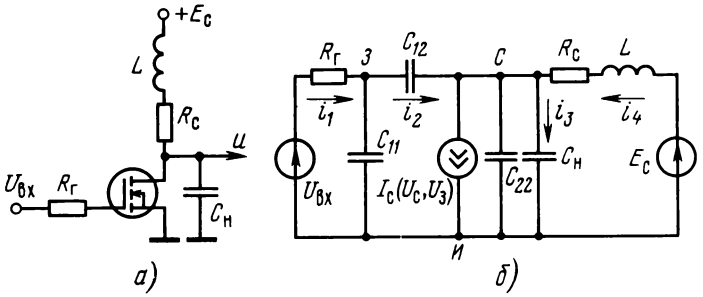


Рис. 7.58

метод сводится к составлению и решению системы дифференциальных уравнений, описывающих работу схемы.

Практическую реализацию метода переменных состояния рассмотрим на примере расчета переходных процессов в каскаде на полевом МДП-транзисторе (рис. 7.58,*а*). На основании эквивалентной схемы (рис. 7.58,*б*) можно записать следующие уравнения:

$$i_1 = \frac{U_{вх} - U_З}{R_и}\,,\quad i_2 = C_0\left[\frac{i_4 - I_С(U_С, U_З)}{C_{22}} - \frac{i_1}{C_{11}}\right];$$

$$i_3 = i_4 - I_С(U_С, U_З) - i_2;$$

$$\frac{1}{C_0} = \frac{1}{C_{11}} + \frac{1}{C_{12}} + \frac{1}{C_{22}};$$

$$\frac{du_З}{dt} = \frac{i_1 + i_2}{C_{11}};\quad \frac{du_С}{dt} = \frac{i_3}{C_{22}};\quad \frac{di_4}{dt} = \frac{E_С - U_С - R_С i_3}{L}.$$

Для придания расчетам большей общности учтем нелинейность емкостей C_{12} и C_{22} в виде зависимостей

$$C_{12} = C_{12\,min} + Ae^{-BU_С},\ C_{22} = C_{22\,min} + Ce^{-DU_С}.$$

Емкость нагрузки можно считать частью $C_{22\,min}$.

Приведенные уравнения при известной зависимости $I_С(U_С, U_З)$ образуют полную динамическую нелинейную модель каскада. Она, как видно, сводится к системе из трех дифференциальных уравнений состояния. Зависимость $I_С(U_С, U_З)$ обычно задается в виде семейства выходных вольт-амперных характеристик, т.е. функций $I_С(U_С)$ для ряда значений $U_З$ = = const. Однако для расчета переходных процессов необходимо вычислять $I_С(U_С, U_З)$ при любых $U_С$ и $U_З$. Таким образом, центральным моментом практической реализации метода переменных состояния является выбор подходящей аппроксимации для зависимости $I_С(U_0, U_З)$: аналитической, полиномиальной и т.д.

Приведенная ниже программа предназначена для совместного использования с программой двухмерной квадратичной сплайн-аппроксимации. Обе программы вводятся в ПЭВМ (например, с кассетного накопителя) и объединяются. Программа сплайн-аппроксимации (см. § 7.10) обеспечивает расчет $I_C(U_C, U_З)$ при любых U_C и $U_З$ на основе только узловых точек ряда кривых $I_C(U_C)$ при заданных $U_З$. Это позволяет оперативно изменять данные $I_C(U_C, U_З)$ и составлять библиотеки семейств выходных вольт-амперных характеристик для различных типов полевых транзисторов.

```
 500 REM МОДЕЛИРОВАНИЕ КАСКАДА Н
А ПОЛЕВОМ ТРАНЗИСТОРЕ С ОБЩИМ ИС
ТОКОМ
 510 REM ВВОД ИСХОДНЫХ ДАННЫХ
 520 INPUT "ВВЕДИТЕ C11 ";C1
 530 INPUT "ВВЕДИТЕ C12 MIN,A,B
";C2,A1,B1
 540 INPUT"ВВЕДИТЕ C22 MIN,C,D "
,C3,C1,D1
 550 INPUT "ВВЕДИТЕ EC,RC,R3,LC
";EC,RC,R3,LC
 560 INPUT "ВВЕДИТЕ UM BX,T BX,T
И ";V1,T1,TI
 570 INPUT "ВВЕДИТЕ HI,M GRAF ";
HI,MG: LET X2=200/MG
 580 LET X1=0: DIM Y(3): DIM A(3
): DIM K(3): DIM W(3): DIM P(3):
 CLS : GO SUB 9600
 585 PRINT AT 21-(10+EC*3)/8,6;"
UC";AT 21-(10+V1*7.5)/8,6;"UI",A
T 19,6;"UG"
 590 PRINT AT 0,2;"MUC=50",AT 0,
10;"MUI,G=20",AT 0,20;"MT=";HI*M
G
 595 LET W(1)=0: LET W(2)=EC: LE
T W(3)=0
 600 REM РЕШЕНИЕ СИСТЕМЫ ДИФФЕРЕ
НЦИАЛЬНЫХ УРАВНЕНИЙ
 610 FOR O=1 TO 3: LET Y(O)=W(O)
: NEXT O: LET I5=0
 620 LET I5=I5+1: GO SUB 800: FO
R O=1 TO 3
 630 LET V=HI*P(O): LET K(O)=V:
LET Y(O)=W(O)+V/2: NEXT O
 640 LET X1=X1+HI/2: GO SUB 800
 650 FOR O=1 TO 3: LET V=HI*P(O)
: LET K(O)=K(O)+V+V
 660 LET Y(O)=W(O)+V/2: NEXT O
 670 GO SUB 800: FOR O=1 TO 3
 680 LET V=HI*P(O): LET K(O)=K(O
)+V+V
 690 LET Y(O)=W(O)+V: NEXT O
 700 LET X1=X1+HI/2: GO SUB 800:
 FOR O=1 TO 3
 710 LET Y(O)=W(O)+(K(O)+HI*P(O)
)/6
 720 REM ВЫВОД ГРАФИКОВ
 730 LET X3=30+I5*X2: PLOT X3,10
+Y(1)*7.5
 740 PLOT X3,Y(2)*3: PLOT X3,10+
U1*7.5
 750 LET W(O)=Y(O): NEXT O
 760 IF X3<=230 THEN GO TO 620
 770 INPUT "КОНЕЦ ВЫЧИСЛЕНИЙ": S
TOP
 800 REM ПОДПРОГРАММА ВЫЧИСЛЕНИЯ
 ПРОИЗВОДНЫХ
 810 LET UI=V1: IF X1<=TI AND T1
<>0 THEN LET UI=V1*(1-EXP (-X1/T
1))
 820 LET U1=UI: IF X1>TI AND T1<
>0 THEN LET U1=UI*EXP (-(X1-TI)/
T1)
 825 IF X1>TI AND T1=0 THEN LET
U1=0
```

```
 830 LET C4=C2: IF A1<>0 THEN LE
T C4=C2+A1*EXP (-B1*Y(2))
 840 LET C5=C3: IF C1<>0 THEN LE
T C5=C3+C1*EXP (-D1*Y(2))
 850 LET I1=(U1-Y(1))/R3: LET C0
=(1/C4+1/C5+1/C1)
 860 LET X=Y(2): LET Y=Y(1): LET
 Z=0: IF Y>3 THEN GO SUB 200
 870 LET I2=((Y(3)-Z)/C5-I1/C1)/
C0
 880 LET I3=Y(3)-Z-I2
 890 LET P(1)=(I1+I2)/C1: LET P(
2)=I3/C5
 900 LET P(3)=(EC-Y(3)*RC-Y(2))/
LC: RETURN
```

Программа построена на основе использования универсальной программы для решения систем из N дифференциальных уравнений (в нашем случае $N=3$), дополненной подпрограммой вычисления производных и фрагментом программы для построения графиков UI $=u_{И}(t)$, UG $=u_{З}(t)$ и UC $=u_{С}(t)$. Ниже даны краткие комментарии к этой программе:

Номера строк	Комментарий
510–570	Ввод исходных данных
590–595	Задание начальных условий и обращение к подпрограмме построения координатных осей графика
600–710	Решение системы из трех дифференциальных уравнений методом Рунге–Кутта
720–770	Вывод по точкам графиков UC, UG и UI
800–900	Подпрограмма вычисления i_1, i_2, i_3 и производных $du_З/dt$, $du_С/dt$ и di_4/dt, а также задания $u_{вх}(t)$, вычисления C_{12}, C_{22} и $I_С(U_С, U_З)$

На рис. 7.59 представлены графики рассчитанных переходных процессов при следующих исходных данных: семейство выходных вольт-амперных характеристик МДП-транзистора соответствует рис. 7.51, $C_{11}=5\cdot10^{-11}$ Ф, $C_{12\,min}=20\cdot10^{-12}$ Ф, $A=B=C=D=0$, $C_{22\,min}=100\cdot10^{-12}$ Ф, $E_С=45$ В, $R_С=50$ Ом, $R_и=50$ Ом, $L_с=200\cdot10^{-9}$ Гн, $U_{вх\,М}=6$ В, $\tau_{вх}=$ ТВХ $=2\cdot10^{-9}$ с, $t_и=$ ТИ $=6\cdot10^{-9}$ с, $h_i=$ HI $=1\cdot10^{-9}$ с (шаг интегрирования), MG $=100$ (число точек каждого графика). Начальные условия: $u_З(0)=0$, $u_С(0)=$

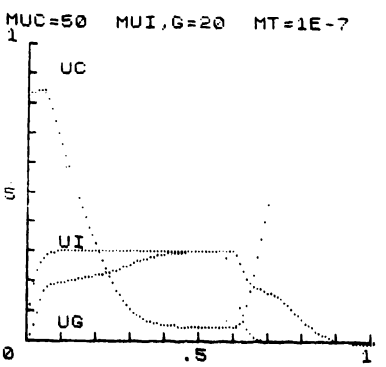


Рис. 7.59

$= E_C$ и $i_4(0) = 0$. Входное воздействие задано в виде импульса длительностью $t_и$ с экспоненциальным нарастанием и спадом:

$$U_{вх}(t) = U_{вх\,М}(1 - e^{-t/\tau_{вх}}) \text{ при } 0 \leqslant t \leqslant t_и,$$

$$U_{вх}(t) = [U_{вх\,М}(1 - e^{-t_и/\tau_{вх}})]\, e^{-(t-t_и)/\tau_{вх}} \text{ при } t > t_и,$$

где $\tau_{вх}$ – постоянная времени нарастания и спада $U_{вх}(t)$.

7.13. СПЕКТРАЛЬНЫЙ АНАЛИЗ СИГНАЛОВ

Сигналы, получаемые с помощью большинства электронных устройств, несинусоидальные. Однако любой реальный периодический сигнал можно представить в виде совокупности гармонических сигналов (гармоник) с частотами, кратными частоте повторения (первой грамоники) f_1 сигнала. Они образуют ряд Фурье

$$y(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \cos 2\pi k f_1 t + b_k \sin 2\pi k f_1 t) = \frac{a_0}{2} +$$

$$+ \sum_{k=0}^{\infty} A_k \cos(2\pi k f_1 t + \varphi_k). \qquad (7.7)$$

Нахождение коэффициентов a_k, b_k или амплитуды A_k и фазы φ_k каждой k-й гармоники составляет цель спектрального анализа. Спектральный анализ имеет важное самостоятельное значение. Например, анализ спектра радиосигналов необходим для оценки полосы передаваемых по радиоканалу частот, обеспечивающей передачу сигналов с допустимым искажением их формы. Анализируя спектры сигналов от ионизационных датчиков, физики могут обнаружить возникновение новых частиц, по спектру токов электротехники могут определить энергетические показатели электротехнических установок и т. д.

Обычно при численном спектральном анализе исходная зависимость $y(t)$ (сигнал) задается рядом значений y_i в моменты времени t_i, разделенные N равными интервалами Δt. При этом коэффициенты a_k, b_k ряда Фурье вычисляются по формулам численного интегрирования методом прямоугольников

$$a_k = \frac{2}{T}\int_0^T y(t)\cos(2\pi k f_1 t)\,dt \approx \frac{2}{N}\sum_{i=1}^{N-1} y_i \cos 2\pi k f_1 i \Delta t, \qquad (7.8)$$

$$b_k = \frac{2}{T}\int_0^T y(t)\sin(2\pi k f_1 t)\,dt \approx \frac{2}{N}\sum_{i=1}^{N-1} y_i \sin 2\pi k f_1 i \Delta t.$$

Амплитуда и фаза гармоники соответственно равны:

$$A_k = \sqrt{a_k^2 + b_k^2},\ \varphi_k = -\operatorname{arctg}(b_k/a_k).$$

Непериодические, но полностью определенные на интервале $t_0 = N\Delta t$ сигналы (финитные) также могут иметь спектральное представление. Но если у периодических сигналов спектр дискретный (т. е. содержит

вполне определенные гармоники с частотами kf_1), то у финитных сигналов спектр сплошной – содержит составляющие с любыми частотами и характеризуется спектральной плотностью $S(f)$ и фазой $\varphi(f)$. Если полагать, что k любые (не обязательно целые) числа, то соотношения (7.8) становятся справедливыми и для непериодических сигналов. В общем случае

$$\frac{a_k N}{2} = \frac{S_c(f)}{\Delta t} \approx \sum_{i=0}^{N-1} y_i \cos 2\pi f i \Delta t,$$
$$\frac{b_k N}{2} = \frac{S_s(f)}{\Delta t} \approx \sum_{i=0}^{N-1} y_i \sin 2\pi f i \Delta t, \qquad (7.9)$$

где $S_c(f)$ и $S_s(f)$ – косинусная и синусная составляющие спектральной плотности $S(f)$. При этом

$$S(f) = \sqrt{S_c^2(f) + S_s^2(f)};$$
$$\varphi(f) = -\operatorname{arctg}[S_s(f)/S_c(f)]. \qquad (7.10)$$

Рассмотрим три основных способа проведения спектрального анализа.

Последовательный спектральный анализ. При последовательном спектральном анализе (7.9) вычисляются по мере ввода y_i для одной частоты f (при периодических $y(t)$, $f = kf_1$, $k = 0, 1, 2, \ldots$), при этом хранить y_i не требуется. Если нужно вычислять спектр, т.е. ряд $S_c(f)$ и $S_s(f)$ или $S(f)$ и $\varphi(f)$ для ряда f, то целесообразно хранить отсчеты y_i в виде одномерного массива. Последовательный спектральный анализ удобен, когда область частот спектра заведомо неизвестна.

В приложении приведена программа П1.10, обеспечивающая выполнение последовательного спектрального анализа с выводом зависимостей $S(f)$ и $\varphi(f)$ в виде графиков, где

Номера строк	Комментарий
10–45	Ввод числа интервалов N, номера первого ненулевого отсчета N1 и последнего M
50–70	Ввод ненулевых отсчетов y_i = Y (I)
80–95	Ввод общего интервала t_0 = T0, числа интервалов спектрограммы K и максимальной частоты спектра f_{max} = FMAKC
100–190	Вычисление $S_c(f)$, $S_s(f)$, $S(f)$ и $\varphi(f)$
200–275	Задание "меню" результатов
280	Вывод таблицы значений y_i = Y (I)
290	Вывод таблицы значений C (F) = S_c и S (F) = S_s
300	Вывод таблицы значений A (F) = S и Q (F) = φ
310–360	Вывод графика Y (T) = $y(t)$ с линейной интерполяцией
370–430	Вывод графика A (F) = $S(f)$
440–490	Вывод графика Q (F) = $\varphi(f)$
800–980	Подпрограммы построения координатных осей графиков и поясняющих надписей

Для контроля программы рассмотрим построение спектрограммы прямоугольного импульса, заданного на $N=32$ интервалах с $\Delta t = 0{,}125 \cdot 10^{-6}$ с ($t_0 = 4 \cdot 10^{-6}$ с), восемью первыми единичными отсчетами и остальными нулевыми. Диалог с ПЭВМ имеет вид:

ВВЕДИТЕ ЧИСЛО ИНТЕРВАЛОВ РАЗБИЕНИЯ N=32
ВВЕДИТЕ НОМЕР ПЕРВОГО НЕНУЛЕВОГО ОТСЧЕТА N1=0
ВВЕДИТЕ НОМЕР ПОСЛЕДНЕГО НЕНУЛЕВОГО ОТСЧЕТА M=7
ВВЕДИТЕ ОТСЧЕТ Y0=1 . . .
ВВЕДИТЕ ОТСЧЕТ Y7=1
ЗАДАЙТЕ ОБЩИЙ ИНТЕРВАЛ T0=4E-6
ВВЕДИТЕ ЧИСЛО ИНТЕРВАЛОВ СПЕКТРА K=20
ЗАДАЙТЕ МАКСИМАЛЬНУЮ ЧАСТОТУ СПЕКТРА FМАКС. 2.5E6
ЗАДАЙТЕ КОД:
0 – ВЫВОД ОТСЧЕТОВ Y (I)
1 – ПОСТРОЕНИЕ ГРАФИКА Y (T)
2 – ВЫВОД ТАБЛИЦЫ C (I) и S (I)
3 – ВЫВОД ТАБЛИЦЫ A (F) и Q (F)
4 – ВЫВОД ГРАФИКА A (F)
5 – ВЫВОД ГРАФИКА Q (F)

На рис. 7.60 показаны графики $Y(T) = y(t)$, $A(F) = S(f)$ и $Q(F) = \varphi(f)$, полученные при задании кодов 1, 4 и 5. Полученные графики дают весьма наглядную картину о спектре прямоугольного импульса.

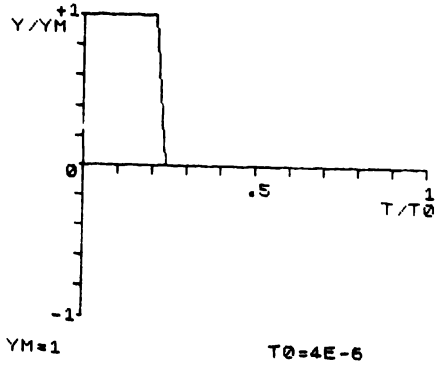
В описанной программе использован прием для повышения точности спектрального анализа. Как известно, при задании $y(t)$ рядом ее значений y_i полученная функция является дискретной. Теоретически показано, что если аппроксимировать $y(t)$ в промежутках между y_i отрезками прямых, то значения $S_c(f)$ и $S_s(f)$ нужно умножить на корректирующий множитель $K1 = (\sin \pi f \Delta t)\,(\pi f \Delta t)^{-1}$. Это сделано в программе и существенно повышает точность построения огибающей спектра.

Для вычисления $\varphi(f)$ в диапазоне углов $\pm\pi$ или $\pm 180^\circ$ используется формула

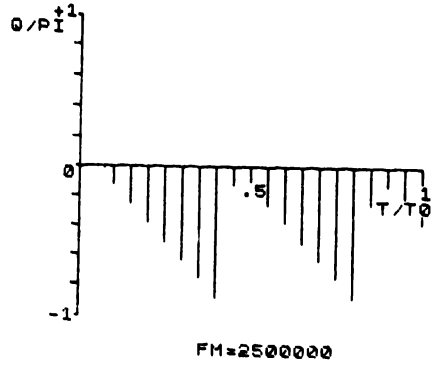
$\varphi(f) = Q = -\arccos(S_c/S_s)$, если $A(f) = R > 1 \cdot 10^{-5}$,
$\varphi(f) = Q = 0$, если $0 < R < 1 \cdot 10^{-5}$,
$\varphi(f) = -Q$, если $S_s < 0$.

При $A(f) = R < 10^{-5}$ результаты вычислений $S_c(f)$ и $S_s(f)$ становятся недостоверными. Чаще всего этот случай наблюдается, когда $S_c(f) = S_s(f) = 0$. Однако на деле из-за присущих ПЭВМ погрешностей получаются малые, но конечные $S_c(f)$ и $S_s(f)$. Вычисление по ним фазы дает неверный результат. Описанная методика автоматически исключает вывод $\varphi(f)$, если $A(f) = R \to 0$, при этом $\varphi(t)$ придаются нулевые значения. Замену $y(t)$ ступенчатой линией ведет к появлению дополнительного фазового сдвига $\Delta\varphi = \pi f \Delta t$. Для коррекции этого сдвига $\Delta\varphi$ вычитается из $\varphi(f) = Q$.

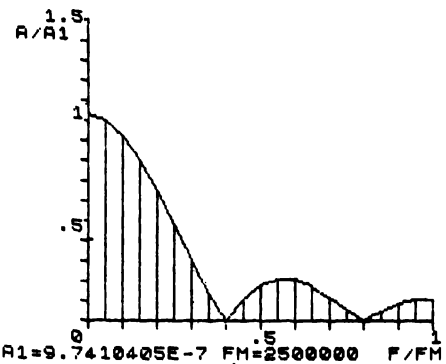
Эти меры существенно повышают точность спектрального анализа. Из рис. 7.60, в частности, видно, что спектр прямоугольного импульса будто снят с картинки учебника [44], хотя спектрограмма получена при



а)



в)



б)

Рис. 7.60

частотах, превышающих частоту второго нулевого спада огибающей спектра. В действительности она чуть не доходит до нуля, что связано с накоплением погрешности построения графика при использовании оператора DRAW и погрешности вычислений $A(f)$.

Графики автоматически масшабируются по оси частот и оси ординат. Масштабирование по оси ординат выполняется относительно значения $S(f)$ = A1 на первой спектральной линии. Масштабные коэффициенты (A1 и FM) выводятся под графиком A/A1 (рис. 7.60, *б*).

Параллельный спектральный анализ. При параллельном спектральном анализе по мере ввода y_i (их можно не хранить) вычисляется заданное число гармоник. Затраты времени при этом те же, что и при последовательном спектральном анализе. Параллельный спектральный анализ удобен, если предположить восстановление $y(t)$ по его данным – тригонометрическая интерполяция $y(t)$. Программы последовательного спектрального анализа нетрудно преобразовать в программы параллельного анализа (программу см. в [1]).

Быстрое преобразование Фурье (БПФ). Вычисление тригонометрических функций по микропрограммам ПЭВМ – процесс относительно длительный. Рассматривая численные значения $\cos 2\pi fi\Delta t$ и $\sin 2\pi fi\Delta t$, нетрудно заметить, что они периодически повторяются. Быстрое преобразование Фурье – это специальный алгоритм, с помощью которого учитывается повторяемость указанных функций.

Дискретное преобразование Фурье (ДПФ). Это специальные преобразования над массивами комплексных чисел. Если задан массив чисел $\dot{u}_j = X_j + iY_j$ $(j = 0, 1, 2, \ldots)$, то прямое ДПФ создает массив чисел

$$\dot{v}_k = \sum_{j=0}^{N-1} u_j \, e^{-i(2\pi kj/N)} .$$

Соответственно, массив чисел $\dot{v}_k$ при обратном ДПФ образует массив

$$u_j = \frac{1}{N} \sum_{k=0}^{N-1} \dot{v}_k \, e^{i(2\pi kj/N)} .$$

Здесь j и k – целые числа.

Программа, выполняющая БПФ на основе ДПФ, дана в приложении 1 [1]. Эта программа может использоваться, если $N = 2^M$ $(M = 1, 2, 3, \ldots)$. Следовательно, число интервалов спектра может выбираться из ряда 2, 4, 8, 16, 32, ... При построении графиков это не совсем удобно. Если задавать $u_j = X_j + iY_j$ действительными числами $(X_j = y_j, Y_j = 0)$, то в результате получим значения S_c и S_s, как и при обычном спектральном анализе, сразу для $N/2 - 1$ гармоники.

Например, для прямоугольного импульса в контрольном примере предшествующей программы, задав $M = 5$ $(N = 32)$, $X_0 = X_1 = \ldots = X_7 = 1$, $X_8 = X_9 = \ldots = X_{32} = 0$ и все $Y_j = 0$ (IS = 0, IF = 7 – номера первого и последнего ненулевых отсчетов), при прямом БПФ получим (указаны первые 5 гармоник):

k	X	Y	M	Q
0	8	8	0,25	0
1	5,57659	−4,57659	0,45088	−39,75
2	1	−5,02734	0,32036	−78,75
3	−1,14828	−2,14828	0,15224	−118,125
4	0	0	0	0
5	1,43543	−1,43543	0,09375	−16,875

Преобразованные X_j, Y_j заносятся на место исходных чисел. Программа выполняет и обратное преобразование Фурье.

ПАКЕТ ПРИКЛАДНЫХ ПРОГРАММ

В приложении приведены листинги наиболее часто используемых программ. В скобках указан тип ПЭВМ, на которой отлаживалась программа.

Программа П1.1 (ДЗ-28)

Численное интегрирование методом Симпсона

```
10 PRINT'ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ МЕТОДОМ СИМПСОНА С ЗАДАННОЙ'
15 PRINT'ТОЧНОСТЬЮ E И F(X) С ОСОБЕННОСТЯМИ ПРИ X=A И X=B'
20 PRINT'':INPUT'ВВЕДИТЕ НИЖНИЙ ПРЕДЕЛ ИНТЕГРИРОВАНИЯ A='A
25 INPUT'ВВЕДИТЕ ВЕРХНИЙ ПРЕДЕЛ ИНТЕГРИРОВАНИЯ B='B
30 INPUT'ВВЕДИТЕ ПОГРЕШНОСТЬ РЕЗУЛЬТАТА E='E:LETM=1
40 PRINT'ВВЕДИТЕ 0 ЕСЛИ НЕТ ОСОБЕННОСТИ ПРИ X=A'
45 INPUT'И 1 ЕСЛИ ЕСТЬ ОСОБЕННОСТЬ ПРИ X=A  'S
50 IF S=0 THEN 65
60 INPUT'ВВЕДИТЕ ЗНАЧЕНИЕ F(A)='C:GOTO 70
65 LETX=A:GOSUB 210:LETC=F
70 PRINT'ВВЕДИТЕ 0 ЕСЛИ НЕТ ОСОБЕННОСТИ ПРИ X=B'
75 INPUT'И 1 ЕСЛИ ЕСТЬ ОСОБЕННОСТЬ ПРИ X=B  'S
80 IF S=0 THEN 100
90 INPUT'ВВЕДИТЕ ЗНАЧЕНИЕ F(B)='D:GOTO 110
100 LETX=B:GOSUB 210:LETD=F:LETJ=0
110 LETM=M*2:LETH=(B-A)/M/2
120 LETN=0:LETI=0:LETX=A
130 LETX=X+H:GOSUB 210
140 LETI=I+4*F:LETN=N+2
150 IF N=2*M THEN 180
160 LETX=X+H:GOSUB 210
170 LETI=I+2*F:GOTO 130
180 LETI=(I+C+D)*H/3:LETK=J:LETJ=I
190 IF ABS(I-K)>E*15 THEN 110
200 PRINT'':PRINT'ЗНАЧЕНИЕ ИНТЕГРАЛА I='I:STOP
205 REM'ПОДПРОГРАММА ВЫЧИСЛЕНИЯ F(X)'
210 LETF=SIN(X)/X:RETURN:END
```

Контрольный пример. Вычисление

$$\mathrm{Si}(Z) = \int_0^Z \frac{\sin x}{x}\,dx.$$

Ввод: $f(a)=1, \varepsilon = 1\cdot 10^4$, $a=\mathrm{A}=0$, $b=\mathrm{B}=Z=1$. Получим $\mathrm{Si}(1)=0{,}9461$.

Программа П1.2 (ДЗ-28)

Решение системы нелинейных уравнений методом Ньютона

```
10 PRINT'РЕШЕНИЕ СИСТЕМЫ НЕЛИНЕЙНЫХ УРАВНЕНИЙ'
20 PRINT'  МОДИФИЦИРОВАННЫМ МЕТОДОМ НЬЮТОНА'
30 INPUT'ЗАДАЙТЕ ЧИСЛО УРАВНЕНИЙ N='N:DIM A(N,N),B(N),X(N),F(N)
40 INPUT'ЗАДАЙТЕ МАКСИМАЛЬНОЕ ЧИСЛО ИТЕРАЦИЙ M='M
```

```
50 INPUT'ЗАДАЙТЕ ОТНОСИТЕЛЬНУЮ ПОГРЕШНОСТЬ E='E:LETS=0
60 FOR I=1 TO N:PRINT!2.0!'ВВЕДИТЕ X'I'(0)'
70 INPUT X(I):NEXT I
80 GOSUB 260:FOR I=1 TO N:LETB(I)=-F(I):NEXT I
90 FOR J=1 TO N:LETX=X(J):LETH=E*ABS(X)
100 LETX(J)=X+H:GOSUB 260:FOR I=1 TO N
110 LETA(I,J)=(F(I)+B(I))/H:NEXT I:LETX(J)=X:NEXT J
120 LETS=S+1:IF S=M+1 THEN PRINT'ЧИСЛО ИТЕРАЦИЙ S=M':STOP
130 FOR I=1 TO N-1:FOR J=I+1 TO N
140 LETA(J,I)=-A(J,I)/A(I,I):FOR K=I+1 TO N
150 LETA(J,K)=A(J,K)+A(J,I)*A(I,K):NEXT K
160 LETB(J)=B(J)+A(J,I)*B(I):NEXT J:NEXT I
170 LETF(N)=B(N)/A(N,N):FOR I=N-1 TO 1 STEP -1
180 LETH=B(I):FOR J=I+1 TO N:LETH=H-F(J)*A(I,J):NEXT J
190 LETF(I)=H/A(I,I):NEXT I:LETR=0
200 FOR I=1 TO N:LETX(I)=X(I)+F(I)
210 IF ABS(F(I)/X(I))>E THEN LETR=1
220 NEXT I:IF R=1 THEN 80
230 PRINT'РЕШЕНИЕ СИСТЕМЫ'
240 FOR I=1 TO N:PRINT!2.0!'X'I!F1.9!'='X(I):NEXT I
245 PRINT!2.0!'ЧИСЛО ИТЕРАЦИЙ S='S:STOP
250 REM'ПОДПРОГРАММА ВЫЧИСЛЕНИЯ F(I)=F(X(1),X(2),...,X(N))'
260 LETF(1)=X(1)+3*LGT(X(1))-X(2)*X(2)
270 LETF(2)=2*X(1)*X(1)-X(1)*X(2)-5*X(1)+1
280 RETURN:END
```

Контрольный пример см. в § 3.2.

Программа П1.3 (ДЗ-28)

Вычисление всех корней полинома

```
05 PRINT'ВЫЧИСЛЕНИЕ ВСЕХ КОРНЕЙ ПОЛИНОМА'
10 PRINT'С ДЕЙСТВИТЕЛЬНЫМИ КОЭФФИЦИЕНТАМИ'
15 INPUT'ЗАДАЙТЕ ПОГРЕШНОСТЬ E='E
20 INPUT'ЗАДАЙТЕ СТЕПЕНЬ ПОЛИНОМА N='N:DIM A(2*N+1)
25 FOR J=1 TO  N+1:PRINT'ВВЕДИТЕ A'!2.0!N+1-J
30 INPUT A(J):NEXT J:LETK=1
35 LETT=1:LETC=A(2)/A(1)
40 IF N=1 THEN LETP=-C:LETQ=0:GOTO 270
50 IF N=2 THEN LETH=C*C/4-A(3)/A(1):GOTO 220
60 LETM=10:LETC=4:LETD=8:LETU=4
70 LETV=8:LETF=1:LETW=2:LETT=0
80 IF M<>10 THEN 100
85 LETP=C:LETM=0:LETQ=D:LETC=U:LETD=V
90 LETU=P:LETV=Q:LETY=C:LETZ=D:LETF=-F
100 LETM=M+1:LETH=0:LETQ=A(1):LETP=A(2)-C*Q:LETL=Q
120 FOR J=3 TO N:LETR=P:LETP=A(J)-C*R-D*Q
130 LETQ=R:LETR=L:LETL=Q-C*R-H*D:LETH=R:NEXT J
140 LETQ=A(N+1)-D*Q:LETS=L+C*R
150 IF T=0 THEN LETX=D*R:LETH=R*X+S*L:IF H=0 THEN 80
160 LETC=C+(P*S-Q*R)/H:LETD=D+(P*X+Q*L)/H
170 IF C-Y+D-Z<>0 THEN 190
180 IF F=-W THEN PRINT'НЕТ РЕШЕНИЯ':STOP
185 LETW=-F
190 LETH=C*C/4-D:IF SQR((Q-P*C/2)^2+P*P*ABS(H))>E/N THEN 80
200 LETT=0:LETA(2)=A(2)-C*A(1):FOR J=3 TO N-1
210 LETA(J)=A(J)-C*A(J-1)-D*A(J-2):NEXT J
220 LETP=-C/2:LETQ=SQR(ABS(H))
230 IF H>=0 THEN LETM=P+Q:LETP=P-Q:LETQ=0:GOTO 250
240 LETM=P
250 PRINT!2.0!'X('K')='!F1.9!M'  +J*('Q')':LETK=K+1
270 PRINT!2.0!'X('K')='!F1.9!P'  +J*('-Q')':LETK=K+1
290 IF T<>0 THEN 350
```

```
300 LETN=N-2:LETA=0
310 IF SQR((S-R*C/2)^2+R*R*ABS(H))<=E THEN LETA=1
320 LETB=0:IF N>=2 THEN LETB=1
330 IF A+B=2 THEN LETT=1:GOTO 110
340  GOTO 35
350 PRINT'ВЫЧИСЛЕНИЯ ЗАКОНЧЕНЫ':END
```

Контрольный пример. Для уравнения $x^4 + 9x^3 + 31x^2 + 60$ при $\varepsilon = 1 \cdot 10^{-6}$ получим $x_1 = -1 + 2, x_2 = -1 - i2, x_3 = -3$ и $x_4 = -4$.

Программа П1.4 (ДЗ-28)

Решение системы дифференциальных уравнений методом Рунге–Кутта

```
10 PRINT'РЕШЕНИЕ СИСТЕМЫ ДИФФЕРЕНЦИАЛЬНЫХ'
20 PRINT'УРАВНЕНИЙ МЕТОДОМ РУНГЕ-КУТТА'
30 INPUT'ВВЕДИТЕ ЧИСЛО УРАВНЕНИЙ N=':N
40 DIM Y(N),A(N),K(N),F(N),W(N)
50 INPUT'ЗАДАЙТЕ ШАГ H=':H
60 INPUT'ЗАДАЙТЕ НАЧАЛЬНОЕ X0=':X
70 FOR J=1 TO N:PRINT!2.0!'ЗАДАЙТЕ НАЧАЛЬНОЕ Y0('J')'
80 INPUT W(J):LETY(J)=W(J):NEXT J
90 GOSUB 200:FOR J=1 TO N:LETV=H*F(J)
100 LETK(J)=V:LETY(J)=W(J)+V/2:NEXT J
110 LETX=X+H/2:GOSUB 200:FOR J=1 TO N
120 LETV=H*F(J):LETK(J)=K(J)+2*V
130 LETY(J)=W(J)+V/2:NEXT J
140 GOSUB 200:FOR J=1 TO N:LETV=H*F(J)
150 LETK(J)=K(J)+2*V:LETY(J)=W(J)+V:NEXT J
160 LETX=X+H/2:PRINT!F1.9!'ДЛЯ X=':X:GOSUB 200
170 FOR J=1 TO N:LETY(J)=W(J)+(K(J)+H*F(J))/6
180 PRINT!2.0!'Y('J')='!F1.9!Y(J)
190 LETW(J)=Y(J):NEXT J:GOTO 90
195 REM ПОДПРОГРАММА ВЫЧИСЛЕНИЯ ПРОИЗВОДНЫХ F(I)
200 LETF(1)=Y(2):LETF(2)=(Y(1)/X-Y(2))/X-Y(1)
210 RETURN:END
```

Контрольный пример. Вычислить y_1 и y_2, решив систему из двух дифференциальных уравнений

$$\frac{dy_1}{dx} = y_2; \quad \frac{dy_2}{dx} = \left(\frac{y_1}{x} - y_2\right)\frac{1}{x} - y_2.$$

С учетом принятых обозначений получим подпрограмму, приведенную в строках 200 и 210. Для $N=2$, $h=0{,}1$, $x_0=0{,}2$, $y_{10} = 0{,}099500833$ и $y_{20}=0{,}49255$ получим:

0,3	0,1483051278	0,4831085139
0,4	0,1960019791	0,4702289086
0,5	0,2422342383	0,453843868
...	...	...

Отметим, что решение данной системы дифференциальных уравнений порождает функцию Бесселя $y_1 = J_1(x)$

Программа П1.5 (ДЗ-28)

Операции с полиномами

```
05 PRINT'ОПЕРАЦИИ С ПОЛИНОМАМИ'
10 INPUT'ЗАДАЙТЕ СТЕПЕНЬ ПОЛИНОМА N=':N:DIM A(3*N):FOR I=0 TO N
20 PRINT !2.0!'ВВЕДИТЕ A'I:DISP 8,1:INPUT A(I):NEXT I
```

```
25 DIM T(3*N):FOR I=0 TO N: T(I)=A(I):NEXT I: V=0
28 FOR I=0 TO N: A(I)=T(I):NEXT I:IF V=1 THEN 52
30  X1=1: X2=1: X3=1: X4=1: X5=1: X6=1:REDL X2='ДИФФ'
40 REDL X3='ИНТЕГР':REDL X4='*':REDL X5='/':REDL X6='P(X)'
45 PRINT'ВВЕДИТЕ ОПЕРАЦИЮ (*, /, ДИФФ, ИНТЕГР, P(X) )'
50  V=1:REDI X1
52 IF X1=X2 THEN 80
54 IF X1=X3 THEN 120
58 IF X1=X4 THEN 160
62 IF X1=X5 THEN 160
66 IF X1=X6 THEN 280
70 PRINT'ТАКУЮ ФУНКЦИЮ НЕЛЬЗЯ ВЫПОЛНИТЬ':END
80 INPUT 'ПОРЯДОК ПРОИЗВОДНОЙ N='L
90 IF L<=N THEN 100
95 PRINT 'F(I)=0':GOTO 28
100 FOR K=1 TO L:FOR I=1 TO N: A(I-1)=A(I)*I:NEXT I:NEXT K
110 FOR I=0 TO N-L:PRINT!2.0!'F'I;!F 1.9!'='A(I):NEXT I:GOTO 28
120 INPUT 'ВЕРХНИЙ ПРЕДЕЛ B='X:INPUT 'НИЖНИЙ ПРЕДЕЛ A='Y
130 FOR I=0 TO N: A(I+1)=A(I)/(I+1): NEXT I
140  A(0)=0: N=N+1:GOSUB 290: B=P: X=Y:GOSUB 290
150  R=ABS(B-P):PRINT!F 1.9!'R='R:GOTO 28
160 INPUT 'СТЕПЕНЬ 2 ПОЛИНОМА 'K:DIM B(3*N)
170 FOR I=0 TO K: PRINT 'ВВЕДИТЕ B'I: DISP 8,1
180 INPUT B(I):NEXT I:IF X1=X5 THEN 230
190 DIM C(N+K):FOR I=N+1 TO N+K: A(I)=0:NEXT I
200 FOR I=K+1 TO N+K: B(I)=0:NEXT I:FOR P=0 TO N+K: C(P)=0
210 FOR I=0 TO P: Q=ABS(P-I): C(P)=C(P)+A(I)*B(Q):NEXT I:NEXT P
220 FOR P=0 TO N+K:PRINT !2.0!'C'P;!F 1.9!'='C(P):NEXT P:END
230 DIM C(N-K):FOR P=N-K TO 0 STEP -1: C(P)=A(P+K)/B(K)
240 FOR D=P TO P+K-1: Q=ABS(D-P): A(D)=A(D)-C(P)*B(Q)
250 NEXT D:NEXT P:PRINT'РЕЗУЛЬТАТ':FOR I=0 TO N-K
260 PRINT!2.0!'C'I;!F 1.9!'='C(I):NEXT I:PRINT'ОСТАТОК'
270 FOR I=0 TO K-1:PRINT!2.0!'A'I;!F 1.9!'='A(I):NEXT I:END
280 INPUT 'X='X:GOSUB 290:PRINT!F 1.9!'P(X)='P:GOTO 280
290  P=A(N):FOR I=N-1 TO 0 STEP -1: P=P*X+A(I):NEXT I:RETURN
```

Это операции ввода коэффициентов a_i и b_i полиномов $A(x)$ и $B(x)$, вычисление $A(x)$ при заданном x, коэффициентов полиномов – производной и неопределенного интеграла $A(x)$, произведения $A(x)B(x)$ и частного $A(x)/B(x)$ с остатком $R(x)$.

Контрольные примеры. Вычисление полиномов $A(x) = 4x^4 + 3x^3 + 2x^2 + 1x + 0$ при $x = 2$ дает $A(2) = 92$, первую производную получаем в виде

$$A'(x) = 16x^3 + 9x^2 + 4x + 1,$$

неопределенный интеграл при $A(x) = 5x^5 + 4x^4 + 3x^3 + 2x^2 + 1x + 1$ в виде

$$A(x) = 0{,}8333x^6 + 0{,}8x^5 + 0{,}75x^4 + 0{,}6667x^3 + 0{,}5x^2 + x.$$

Для полиномов $A(x) = 4x^4 + 3x^3 + 2x^2 + x + 0{,}5$ и $B(x) = 2x^2 + 1x + 0{,}4$ получим

$$A(x)B(x) = 8x^6 + 10x^5 + 8{,}6x^4 + 5{,}2x^3 + 2{,}8x^2 + 0{,}9x + 0{,}2.$$

Деление полиномов можно проверить по следующему примеру:

$$\frac{A(x)}{B(x)} = \frac{1x^6 + 7x^5 + 15x^4 + 15x^3 + 4x^2 - 3x + 125}{x^2 - 2x + 2} = 1x^4 + 9x^3 + 31x^2 + 59x + 60$$

при остатке $R(x) = -1x + 5$.

Универсальная программа построения графиков функций и их аппроксимации (UGD)

```
  10 CLS : PRINT AT 1,11;"***UGD
***";AT 2,4;"УНИВЕРСАЛЬНАЯ ПРОГР
АММА";AT 3,5;"ПОСТРОЕНИЯ ГРАФИКО
В";AT 5,11;"МЕНЮ f(X):"
  15 PRINT AT 7,2;"1-АНАЛИТИЧЕСК
АЯ ";AT 8,2;"2-ЗАДАЕТСЯ ПОДПРОГР
АММОЙ";AT 9,2;"3-АППРОКСИМИРУЕТС
Я ПОЛИНОМОМ";AT 10,2;"4-АППРОКСИ
МИРУЕТСЯ ПОЛИНОМОМ";AT 11,2;"  М
ЕТОДОМ НАИМЕНЬШИХ КВАДРАТОВ"
  20 PRINT AT 12,2;"5-СПЛАЙН-АПП
РОКСИМАЦИЯ ГЛОБАЛ.";AT 13,2;"6-С
ПЛАЙН-АППРОКСИМАЦИЯ ЛОКАЛ.";AT 1
4,2;"7-МНОГОИНТЕРВАЛЬНАЯ КВАДРАТ
И-";AT 15,2;"  ЧНАЯ АППРОКСИМАЦИ
Я";AT 16,2;"8-ЧЕБЫШЕВСКАЯ АППРОК
СИМАЦИЯ"
  30 INPUT AT 0,0;"f(x):1-АНАЛ 2
-ПП 3-ПОЛ 4-НК 5-СГ 6-СЛ 7-МК 8-
ЧЕБ ";N2
  35 IF N2<1 OR N2>8 THEN GO TO
50
  40 IF N2=1 OR N2=2 THEN GO SUB
 1000+100*N2
  42 IF N2=3 OR N2=5 OR N2=6 OR
N2=7 THEN LET N1=N4
  45 IF N2>=3 AND N2<8 THEN GO S
UB 1300
  50 IF n2=8 THEN GO SUB 1500
  55 IF N2>=3 AND N2<=7 THEN GO
SUB 2000+100*N2
  60 INPUT AT 0,0;"ГРАФИК-G, НАЛ
ОЖЕНИЕ-N ВЫЧИСЛЕНИЕ f(x)-C ? ";y
$
  65 IF y$="n" OR y$="N" THEN GO
 TO 110
  70 IF y$="g" OR y$="G" THEN CL
S : GO SUB 120: GO TO 110
  80 CLS : IF N2=3 OR N2=4 THEN
CLS : PRINT "ЗАДАН МАССИВ:": F
OR I=1 TO N4: PRINT "X";i-1;"=";
X(I),"y";I-1;"=";y(I): NEXT I: P
RINT ''"КОЭФФИЦИЕНТЫ ПОЛИНОМА:":
 FOR I=1 TO N1: PRINT "a";I-1;
"=";Z(I): NEXT I: PRINT
  85 IF N2=5 OR N2=6 THEN CLS :
PRINT "ЗАДАН МАССИВ:": FOR I=1
 TO N1: PRINT "x";I-1;"=";x(I),"
y";I-1;"=";y(I): NEXT I: PRINT '
"КОЭФФИЦИЕНТЫ СПЛАЙН-ФУНКЦИИ:":
 FOR I=1 TO N1: PRINT "m";I-1;"=
";m(I): NEXT I: PRINT
  90 IF N2=7 THEN CLS : PRINT '"
МАССИВ УЗЛОВ:": FOR I=1 TO N1:
PRINT "X",I-1;"=";x(I),"y";I-1;"
=";y(I): NEXT I: PRINT
  95 IF N2=8 THEN CLS : PRINT "M
АССИВ УЗЛОВ:": FOR I=1 TO Q: PRI
NT "X";I-1;"=";X(I),"Y";I-1;"=";
Y(I): NEXT I: PRINT '"КОЭФФИЦИЕН
ТЫ ПОЛИНОМА ЧЕБЫШЕВА:": FOR I=1
 TO Q: PRINT "a";I-1;"=";a(i): N
EXT I  PRINT
 100 INPUT "ВВЕДИТЕ x,M(МЕНЮ),G(
ГРАФИК) ";X$: IF X$="m" OR X$="M
" THEN CLS : GO TO 10
 102 IF X$="G" OR X$="g" THEN CL
S : GO SUB 120: GO TO 110
 105 LET X=VAL X$: GO SUB 3000+1
00*N2: PRINT "f(";x;")=";f: GO T
O 100
 110 GO SUB 410: PAUSE 0: GO TO
30
 120 INPUT "ВВЕДИТЕ Xmin=",xmin;
"  xmax=";xmax
 130 LET ymax=75: LET ymin=-75
 140 INPUT "ВВОДИТЬ ПРЕДЕЛЫ Y (Y
/N)? ",y$
```

```
150 IF y$="n" OR y$="N" THEN GO
TO 200
160 INPUT "ВВЕДИТЕ Ymin=" ymin;
" И Ymax=";ymax
170 IF ymin>=ymax THEN GO TO 16
0
200 REM КООРДИНАТНЫЕ ОСИ
230 LET xrange=xmax-xmin  LET y
range=ymax-ymin  LET dx=xrange/2
00  LET dy=yrange/144
250 LET cy=ABS (ymin)*(ymin<0)+
LET cy=cy/dy+16
260 LET cx=ABS (xmin)*(xmin<0):
LET cx=cx/dx+32
280 PLOT 32,cy: DRAW 200,0
290 PLOT cx,16: DRAW 0,144
300 REM alculate scale values a
nd print on axes
310 LET yscale=INT ((175-cy)/8)
+1
320 LET xscale=INT (cx/8)-4
330 FOR n=0 TO 5
340 LET sx=(INT (10*(xmin+n*xra
nge/5)+.5))/10
350 PRINT OVER 1;AT yscale,5*n+
4;sx  PLOT (5*n+4)*8,cy-1: NEXT
n
360 FOR n=0 TO 6
370 LET sy=(INT (10*(ymax-n*yra
nge/6)+.5))/10
380 LET g$=STR$ sy
390 IF LEN g$<4 THEN FOR w=LEN
g$ TO 3: LET g$=" "+g$: NEXT w
400 PRINT OVER 1;AT 3*n+1,xscal
e;g$: PLOT cx+1,(3*n+2)*8: NEXT
n
405 RETURN
410 REM ГРАФИК ФУНКЦИИ
415 LET x=xmin: GO SUB 3000+100
*N2  LET y0=(f-ymin)/dy
420 LET y1=-1: FOR n=0 TO 200
430 LET x=xmin+n*dx+1e-6: GO SU
B 3000+100*n2
440 LET y=(f-ymin)/dy: IF y>=15
8 THEN LET y=158: LET y0=y: GO T
O 460
450 LET y1=y: IF y1<-14 THEN LE
T y1=-14: LET y0=-14
460 PLOT n+31,y0+16: DRAW 1,y1-
y0: LET y0=y0+INT ((y1-y0)+.5)
470 NEXT N: IF N2=1 THEN RETURN
475 INPUT "МЕТИТЬ УЗЛЫ (Y/N)? "
;Y$: IF Y$="N" OR Y$="n" THEN RE
TURN
480 FOR I=1 TO N4: CIRCLE 32+(x
(I)-xmin)/dx,16+(y(I)-ymin)/dy,1
.4: NEXT I: RETURN
1000 REM ВВОД ИСХОДНЫХ ДАННЫХ
1100 INPUT "F(x)=";f$: RETURN
1200 REM ЗАДАЙТЕ ПОДПРОГРАММУ F(
x) СО СТРОКИ 3200
1210 RETURN
1300 REM ЗАДАНИЕ МАССИВОВ X(I),Y
(I)
1302 INPUT "ВВОДИТЬ ДАННЫЕ (Y/N)
? ";Y$: IF Y$="N" OR Y$="n" THEN
 RETURN
1305 INPUT "ВВЕДИТЕ НОМЕР ПОСЛЕД
НЕГО ОТСЧЕТА N=";N1: LET N1=N1+1
: LET N4=N1
1310 DIM A(N1,N1): DIM B(N1)  DI
M C(2*N1): DIM Z(N1)
1315 DIM X(N1): DIM Y(N1): DIM L
(N1): DIM M(N1): DIM R(N1): DIM
S(N1)
1320 INPUT "РАВНОМЕРНО ЛИ РАСПОЛ
ОЖЕНЫ УЗЛЫ (Y/N)? ";Y$
1330 IF Y$="n" OR Y$="N" THEN GO
 TO 1360
1340 INPUT "ВВЕДИТЕ ЗНАЧЕНИЕ X0=
";X0;"  И ШАГ H=";H
```

```
1350 FOR I=1 TO N1: LET X(I)=X0+
(I-1)*H: INPUT "ВВЕДИТЕ Y";(I-1)
;"=";Y(I): NEXT I: GO TO 1370
1360 FOR I=1 TO N1: INPUT "ВВЕДИ
ТЕ X";(I-1);"=";X(I);"  И Y";(I-
1);"=";Y(I): NEXT I
1370 RETURN
1500 REM ВЫЧИСЛЕНИЕ КОЭФФИЦИЕНТО
В МНОГОЧЛЕНА ЧЕБЫШЕВА T(I)=A(I)
1510 INPUT "ВВЕДИТЕ СТЕПЕНЬ МНОГ
ОЧЛЕНА M=";N1
1520 LET N4=N1+1: LET Q=N4: DIM
X(Q): DIM Y(Q): DIM A(Q)
1530 INPUT "ВВЕДИТЕ ГРАНИЦЫ X  A
=";A1;"  И B=";B1
1540 LET M=PI/2/Q: LET C=(B1+A1)
/2: LET D=(B1-A1)/2
1550 INPUT "f(x) ЗАДАНА ПОДПРОГР
АММОЙ (Y/N)? ";Y$
1560 LET V=0: IF Y$="N" OR Y$="n
" THEN LET V=1
1570 FOR I=0 TO N1: LET X1=COS (
(2*I+1)*M): LET X=C+D*X1: LET X(
I+1)=X
1580 IF V=1 THEN LET f=VAL f$
1590 IF V=0 THEN GO SUB 3200
1600 LET y(i+1)=f: NEXT I
1610 FOR K=0 TO N1: FOR I=0 TO N
1
1620 LET A(K+1)=A(K+1)+Y(I+1)*CO
S (K*(2*I+1)*M): NEXT I
1630 LET A(K+1)=A(K+1)*2/Q: NEXT
 K: GO SUB 2800: RETURN
2000 REM ВЫЧИСЛЕНИЕ КОЭФФИЦИЕНТО
В ПОЛИНОМОВ И СПЛАЙНОВ
2300 REM ВЫЧИСЛЕНИЕ КОЭФФИЦИЕНТО
В ПОЛИНОМА
2310 DIM A(N1,N1): DIM B(N1): DI
M Z(N1)
2320 FOR I=1 TO N1: LET B(I)=Y(I
): LET A(I,1)=1: LET R=1
2330 FOR J=2 TO N1: LET R=R*X(I)
: LET A(I,J)=R: NEXT J: NEXT I
2340 FOR I=1 TO N1-1: FOR J=I+1
TO N1
2350 LET A(J,I)=-A(J,I)/A(I,I)
2360 FOR K=I+1 TO N1: LET A(J,K)
=A(J,K)+A(J,I)*A(I,K): NEXT K
2370 LET B(J)=B(J)+A(J,I)*B(I):
NEXT J: NEXT I
2375 LET Z(N1)=B(N1)/A(N1,N1)
2380 FOR I=N1-1 TO 1 STEP -1: LE
T H1=B(I)
2385 FOR J=I+1 TO N1: LET H1=H1-
Z(J)*A(I,J): NEXT J
2390 LET Z(I)=H1/A(I,I): NEXT I
2395 RETURN
2400 REM ОБРАБОТКА ДАННЫХ МНК
2410 LET N3=N1: INPUT "ЗАДАЙТЕ С
ТЕПЕНЬ ПОЛИНОМА M=";N1: LET N1=N
1+1
2420 FOR I=1 TO N3: LET Y=Y(I):
LET X=X(I): LET F=1
2430 FOR J=1 TO 2*N1-1: IF J>N1
THEN GO TO 2450
2440 LET B(J)=B(J)+Y: LET Y=Y*X
2450 LET C(J)=C(J)+F: LET F=F*X:
 NEXT J: NEXT I
2460 FOR I=1 TO N1: LET K=I: FOR
 J=1 TO N1
2470 LET A(I,J)=C(K): LET K=K+1:
 NEXT J: NEXT I: GO TO 2320
2500 REM ВЫЧИСЛЕНИЕ M(I) ГЛОБАЛЬ
НОГО СПЛАЙНА
2510 LET d=x(2)-x(1): LET e=(y(2
)-y(1))/d
2520 FOR k=2 TO n1-1: LET h=d: L
ET d=x(k+1)-x(k)
2525 LET f=e: LET e=(y(k+1)-y(k)
)/d: LET l(k)=d/(d+h)
2530 LET r(k)=1-l(k): LET s(k)=6
*(e-f)/(h+d): NEXT k
```

```
2540 FOR k=2 TO n1-1: LET p=1/(r
(k)*l(k-1)+2): LET l(k)=-l(k)*p
2550 LET s(k)=(s(k)-r(k)*s(k-1))
*p: NEXT k
2560 LET m(n1)=0: LET l(n1-1)=s(
n1-1): LET m(n1-1)=l(n1-1)
2570 FOR k=n1-2 TO 1 STEP -1
2580 LET l(k)=l(k)*l(k+1)+s(k):
LET m(k)=l(k): NEXT k
2590 RETURN
2600 REM ЗАДАНИЕ ИЛИ ВЫЧИСЛЕНИЕ
M(I) ЛОКАЛЬНОГО СПЛАЙНА
2610 INPUT "БУДЕТЕ ЗАДАВАТЬ M(I)
-(Y/N) ";Y$
2620 IF Y$="N" OR Y$="n" THEN GO
 TO 2640
2630 FOR I=1 TO N1: INPUT "ВВЕДИ
TE M";(I-1);"=";M(I): NEXT I: RE
TURN
2640 LET M(1)=(4*Y(2)-Y(3)-3*Y(1
))/2/H
2650 LET M(N1)=(3*Y(N1)+Y(N1-2)-
4*Y(N1-1))/2/H
2660 FOR I=2 TO N1-1: LET M(I)=(
Y(I+1)-Y(I-1))/2/H
2670 NEXT I: RETURN
2700 RETURN
2800 REM ПРЕОБРАЗОВАНИЕ МНОГОЧЛЕ
НА T(I) В P(I)
2810 LET A(1)=A(1)/2: FOR J=2 TO
 N1: FOR K=N1 TO J STEP -1
2820 LET A(K-1)=A(K-1)-A(K+1): L
ET A(K+1)=2*A(K+1)
2830 NEXT K: NEXT J: RETURN
3000 REM ВЫЧИСЛЕНИЕ F=f(X)
3100 LET f=VAL F$: RETURN
3200 REM ПОДПРОГРАММА ВЫЧИСЛЕНИЯ
 f=f(x)
3210 LET f=SIN x/x
3299 RETURN
3300 REM ВЫЧИСЛЕНИЕ ЗНАЧЕНИЙ ПОЛ
ИНОМА
3310 LET F=0: FOR I=N1 TO 2 STEP
 -1
3320 LET F=(F+Z(I))*X: NEXT I
3330 LET F=F+Z(1): RETURN
3400 GO TO 3300
3500 REM ВЫЧИСЛЕНИЕ F(X) ГЛОБАЛЬ
НОГО СПЛАЙНА
3510 LET i=0: IF x>x(n1) THEN GO
 TO 3575
3520 IF x<=x(1) THEN GO TO 3585
3530 LET i=i+1: IF x>x(i) THEN G
O TO 3530
3550 LET j=i-1: LET d=x(i)-x(j):
 LET h1=x-x(j): LET r=x(i)-x
3560 LET p=d*d/6: LET f=(m(j)*r*
r*r+m(i)*h1*h1*h1)/6/d
3570 LET f=f+((y(j)-m(j)*p)*r+(y
(i)-m(i)*p)*h1)/d: RETURN
3575 LET d=x(n1)-x(n1-1): LET f=
d*m(n1-1)/6+(y(n1)-y(n1-1))/d
3580 LET f=f*(x-x(n1))+y(n1): RE
TURN
3585 LET d=x(2)-x(1): LET f=-d*m
(2)/6+(y(2)-y(1))/d
3590 LET f=f*(x-x(1))+y(1): RETU
RN
3600 REM ВЫЧИСЛЕНИЕ ГЛОБАЛЬНОГО
СПЛАЙНА
3610 IF X<=X(1) THEN GO TO 3680
3620 IF X>=X(N1) THEN GO TO 3690
3630 LET I=INT ((X-X(1))/H): LET
 B=X(1)+H*I
3640 LET C=B+H: LET D=X-C: LET E
=X-B
3650 LET f=D*D*(E+E+H)*Y(I+1)+E*
E*(H-D-D)*Y(I+2)
3660 LET f=f/H+D*E*M(I+1)
3670 LET f=(f+E*E*D*M(I+2))/H/H:
 RETURN
3680 LET f=Y(1)+(X-X(1))*M(1): R
ETURN
```

```
3690 LET f=Y(N1)+(X-X(N1))*M(N1)
: RETURN
3700 REM ВЫЧИСЛЕНИЕ f(x) ПРИ МНО
ГОИНТЕРВАЛЬНОЙ КВАДРАТИЧНОЙ АППР
ОКСИМАЦИИ
3710 IF x<=x(1) THEN GO TO 3760
3720 IF x>=x(n1) THEN GO TO 3770
3730 LET i=INT ((x-x(1))/h): IF
i>=n1-2 THEN LET i=n1-2
3735 IF i=0 THEN LET i=1
3740 LET p=(x-x(1)-i*h)/h: LET f
=p*(p-1)*y(i)/2+(1-p*p)*y(i+1)+p
*(p+1)*y(i+2)/2: RETURN
3750 IF i=n1 THEN RETURN
3760 LET f=y(1)+(x-x(1))*(4*y(2)
-y(3)-3*y(1))/2/h: RETURN
3770 LET f=y(n1)+(x-x(n1))*(3*y(
n1)+y(n1-2)-4*y(n1-1))/2/h: RETU
RN
3800 REM ВЫЧИСЛЕНИЕ f(x) ПО МНОГ
ОЧЛЕНУ ЧЕБЫШЕВА
3810 LET f=0: LET X1=X: LET X=(X
1-C)/D
3820 FOR I=N1 TO 1 STEP -1: LET
f=(f+A(I+1))*X: NEXT I
3830 LET x=x1: LET f=f+A(1): RET
URN
```

Контрольные примеры см. в § 7.11.

Программа П1.7 (ZX-Spectrum)

Регрессия для 16 видов зависимостей $y(x)$ (R16)

```
   1 REM РЕГРЕССИЯ ДЛЯ 16 Y(X)
   5 CLS : PRINT "  РЕГРЕССИЯ ДЛ
Я 16 Y(X)": LET L=LN 10
  10 PRINT AT 2,1;"МЕНЮ ФУНКЦИЙ:
";" 1 Y=A+B*X";" 2 Y=1/(A+B*X)";
" 3 Y=A+B/X";" 4 Y=X/(A+B*X)";"
5 Y=A*B↑X";" 6 Y=A*EXP(B*X)";" 7
 Y=A*10↑(B*X)";" 8 Y=1/(A+B*EXP(
-X))";" 9 Y=A*X↑B";" 10 Y=A+B*LO
G(X)";" 11 Y=A+B*LN(X)";" 12 Y=A
/(B+X)";" 13 Y=A*X/(B+X)";" 14 Y
=A*EXP(B/X)";" 15 Y=A*10↑(B/X)";
" 16 Y=A+B*(X↑N)"
  15 INPUT "БУДЕТЕ ВВОДИТЬ ДАННЫ
Е (Y/N) ";Y$
  20 IF Y$="N" OR Y$="n" THEN GO
 TO 50
  25 LET M=100: DIM X(M): DIM Y(
M): LET J=0
  30 LET J=J+1: INPUT "ВВЕДИТЕ X
";(J);"=";X$;" Y";(J);"=";Y$
  35 IF X$="N" OR Y$="N" OR X$="
n" OR Y$="n" THEN LET J=J-1: GO
TO 50
  40 LET X(J)=VAL X$: LET Y(J)=V
AL Y$
  45 GO TO 30
  50 INPUT "ЗАДАЙТЕ НОМЕР ФУНКЦИ
И ";K
  55 IF K=16 THEN INPUT "ВВЕДИТЕ
 N=";N9
  60 GO SUB 1000: INPUT "БУДЕТЕ
СМОТРЕТЬ ЧИСЛОВЫЕ ДАННЫЕ (Y/N) "
;Y$
  65 IF Y$="N" OR Y$="n" THEN CL
S : GO TO 85
  70 CLS : PRINT AT 0,2;"МАССИВ
ИСХОДНЫХ ДАННЫХ": FOR I=1 TO J:
 PRINT "X";I;"=";X(I),"Y";I;"=";
Y(I): NEXT I: PRINT "НАЖМИТЕ ЛЮБ
УЮ КЛАВИШУ!": PAUSE 0
  75 CLS : PRINT "K=";K'"A=";A'"
B=";B'"R=";R
  80 INPUT "ВВЕДИТЕ X(ИЛИ N) ";X
$: IF X$="N" OR X$="n" THEN GO T
O 85
```

```
  82 LET X=VAL X$: GO SUB 2400+1
0*K: PRINT "Y(";Xi")=";Y: GO TO
80
  85 INPUT "ГРАФИК(G), НАЛОЖЕНИЕ
(N), МЕНЮ(M) ?";Y$
  90 IF Y$="N" OR Y$="n" THEN IN
PUT "ВВЕДИТЕ НОМЕР ФУНКЦИИ (ИЛИ
0 ПРИ ОКОНЧАНИИ) ";K: IF K=0 THE
N STOP
  92 IF Y$="N" OR Y$="n" THEN GO
 SUB 1000: GO SUB 410: GO TO 90
  95 IF Y$="G" OR Y$="g" THEN GO
 SUB 110: GO SUB 410: GO TO 85
 100 IF Y$="M" OR Y$="m" THEN GO
 TO 5
 105 GO TO 85
 110 CLS : INPUT "ВВЕДИТЕ Xmin="
,xmin;" Xmax=";xmax
 120 IF xmin>=xmax THEN GO TO 11
0
 130 LET ymax=75: LET ymin=-75
 140 INPUT "БУДЕТЕ ВВОДИТЬ ПРЕДЕ
ЛЫ Y (Y/N) ? ";y$
 150 IF y$="n" OR y$="N" THEN GO
 TO 200
 160 INPUT "ВВЕДИТЕ Ymin=";ymin;
" Ymax=";ymax
 170 IF ymin>=ymax THEN GO TO 16
0
 230 LET xrange=xmax-xmin: LET y
range=ymax-ymin: LET dx=xrange/2
00: LET dy=yrange/144
 250 LET cy=ABS (ymin)*(ymin<0):
 LET cy=cy/dy+16
 260 LET cx=ABS (xmin)*(xmin<0):
 LET cx=cx/dx+32
 280 PLOT 32,cy: DRAW 200,0
 290 PLOT cx,16: DRAW 0,144
 300 REM alculate scale values a
nd print on axes
 310 LET yscale=INT ((175-cy)/8)
+1
 320 LET xscale=INT (cx/8)-4
 330 FOR n=0 TO 5
 340 LET sx=(INT (10*(xmin+n*xra
nge/5)+.5))/10
 350 PRINT OVER 1;AT yscale,5*n+
4;sx: PLOT (5*n+4)*8,cy-1: NEXT
n
 360 FOR n=0 TO 6
 370 LET sy=(INT (10*(ymax-n*yra
nge/6)+.5))/10
 380 LET g$=STR$ sy
 390 IF LEN g$<4 THEN FOR c=LEN
g$ TO 3: LET g$=" "+g$: NEXT c
 400 PRINT OVER 1;AT 3*n+1,x$cal
e;g$: PLOT cx+1,(3*n+2)*8: NEXT
n: RETURN
 410 REM ТОЧКИ Y(X)
 415 LET x=xmin+1e-6: GO SUB 240
0+10*k: LET y0=(y-ymin)/dy
 420 FOR n=1 TO 200
 430 LET x=xmin+n*dx: GO SUB 240
0+10*K
 440 LET y1=(y-ymin)/dy: IF y1>=
158 THEN LET y1=158: LET y0=y1:
GO TO 460
 450 IF y1<-15 THEN LET y1=-15:
LET y0=y1: GO TO 460
 460 PLOT n+31,y0+16: DRAW 1,y1-
y0: LET y0=y0+INT ((y1-y0)+.5)
 470 NEXT n
 480 INPUT "БУДЕТЕ ОТМЕЧАТЬ УЗЛЫ
 (Y/N) ";Y$: IF Y$="N" OR Y$="n"
 THEN PAUSE 0: RETURN
 490 FOR I=1 TO J: CIRCLE 32+(X(
I)-xmin)/dx,16+(Y(I)-ymin)/dy,1.
45: NEXT I: PAUSE 0: RETURN
1000 REM РЕГРЕССИЯ
1010 LET X1=0: LET X2=0: LET Y1=
0: LET Y2=0: LET P=0
1020 FOR I=1 TO J: LET X=X(I): L
ET Y=Y(I): GO TO 1200+10*K
```

```
1030 LET A=W: LET B=V: GO TO 200
0+10*K
1200 REM ЛИНЕАРИЗУЮЩИЕ ПРЕОБРАЗО
ВАНИЯ
1210 GO TO 1600
1220 LET Y=1/Y: GO TO 1600
1230 LET X=1/X: GO TO 1600
1240 LET Y=X/Y: GO TO 1600
1250 LET Y=LN Y/L: GO TO 1600
1260 LET Y=LN Y: GO TO 1600
1270 LET Y=LN Y/L: GO TO 1600
1280 LET X=EXP (-X): LET Y=1/Y:
GO TO 1600
1290 LET X=LN (X)/L: LET Y=LN Y/
L: GO TO 1600
1300 LET X=LN X/L: GO TO 1600
1310 LET X=LN X: GO TO 1600
1320 LET Y=1/Y: GO TO 1600
1330 LET X=1/X: LET Y=1/Y: GO TO
 1600
1340 LET X=1/X: LET Y=LN Y: GO T
O 1600
1350 LET X=1/X: LET Y=LN Y/L: GO
 TO 1600
1360 LET X=X↑N9: GO TO 1600
1600 REM ЛИНЕЙНАЯ РЕГРЕССИЯ
1610 LET X1=X1+X: LET Y1=Y1+Y: L
ET X2=X2+X*X
1620 LET Y2=Y2+Y*Y: LET P=P+X*Y:
 NEXT I
1630 LET V=(X1*Y1-J*P)/(X1*X1-J*
X2)
1640 LET W=(Y1-V*X1)/J
1650 LET R=(P-X1*Y1/J)/SQR ((X2-
X1*X1/J)*(Y2-Y1*Y1/J))
1660 GO TO 1030
2000 REM СВЕДЕНИЕ НЕЛИНЕЙНОЙ РЕГ
РЕССИИ К ЛИНЕЙНОЙ
2040 RETURN
2050 LET A=10↑W: LET B=10↑V: RET
URN
2060 LET A=EXP W: RETURN
2070 LET A=10↑W: RETURN
2080 RETURN
2090 GO TO 2070
2110 RETURN
2120 LET A=1/V: LET B=W/V: RETUR
N
2130 LET A=1/W: LET B=V/W: RETUR
N
2140 GO TO 2060
2150 GO TO 2070
2160 RETURN
2400 REM ВЫЧИСЛЕНИЕ Y(X)
2410 LET Y=A+B*X: RETURN
2420 LET Y=1/(A+B*X): RETURN
2430 LET Y=A+B/X: RETURN
2440 LET Y=X/(A+B*X): RETURN
2450 LET Y=A*B↑X: RETURN
2460 LET Y=A*EXP (B*X): RETURN
2470 LET Y=A*10↑(B*X): RETURN
2480 LET Y=1/(A+B*EXP (-X)): RET
URN
2490 LET Y=A*X↑B: RETURN
2500 LET Y=A+B*LN X/L: RETURN
2510 LET Y=A+B*LN X: RETURN
2520 LET Y=A/(B+X): RETURN
2530 LET Y=A*X/(B+X): RETURN
2540 LET Y=A*EXP (B/X): RETURN
2550 LET Y=A*10↑(B/X): RETURN
2560 LET Y=A+B*X↑N9: RETURN
```

Контрольные примеры см. в § 7.11.

Программа П1.8 (ZX-Spectrum)

Анализ линейных лестничных цепей с топологическим вводом данных

```
  10 PRINT "     АНАЛИЗ ЛЕСТНИЧНЫ
Х ЦЕПЕЙ""  МЕТОДОМ ДИСКРЕТНЫХ М
ОДЕЛЕЙ"
  20 PRINT ''"ВНИМАНИЕ!  НУМЕРАЦИ
Я ВЕТВЕЙ НАЧИНАЕТСЯ С ПОСЛЕДНЕЙ
ВЕТВИ.ДОПУСКАЕТСЯ ПАРАЛЛЕЛЬНОЕ С
```

```
ОЕДИНЕНИЕ R,C,L В ЛЮБОЙ ВЕТВИ.ПР
И ПОСЛЕДОВАТЕЛЬНОМ СОЕДИНЕНИИ В
ГОРИЗОНТАЛЬНОЙ ВЕТВИ К ТОЧКЕ СОЕ
ДИНЕНИЯ ВКЛЮЧАЕТСЯ ПРОНУМЕРОВАНН
АЯ ВЕРТИКАЛЬНАЯ ВЕТВЬ С GI=0"
  30 PRINT " ЗАПИСЬ ТОПОЛОГИЧЕСК
ИХ МАССИВОВ:"
  40 PRINT "G,I,NOM-МАССИВ ПРОВО
ДИМОСТЕЙ","C,I,NOM-МАССИВ ЕМКОСТ
ЕЙ","L,I,NOM-МАССИВ ИНДУКТИВНОСТ
ЕЙ","F,A,ω,ТИ-МАССИВ ВОЗДЕЙСТВИ
Й"
  50 PRINT " ГДЕ I-НОМЕР ВЕТВИ,A
-АМПЛИТУДА,W-ЧАСТОТА,ω-НАЧАЛЬНА
Я ФАЗА,ТИ-ДЛИТЕЛЬНОСТЬ ВОЗДЕЙСТВ
ИЯ"''" НАЖМИТЕ ЛЮБУЮ КЛАВИШУ": P
AUSE 0: CLS
  60 PRINT ''"      ВИДЫ ВОЗДЕЙСТ
ВИЙ:"''"1.W=0,ω=0,ТИ<ТК - ПРЯМО
УГОЛЬНЫЙ ИМПУЛЬС С ДЛИТЕЛЬНОСТЬЮ
 ТИ"''"2.W=0,ω=0,ТИ=ТК - ПЕРЕПА
Д"''"3.W=0,ω=0,ТИ<0 - ЭКСПОНЕНЦ
ИАЛЬНЫЙ ПЕРЕПАД F=A*(1-EXP(t/ТИ)
)"''"4.W<>0,ω,ТИ - ГАРМОНИЧЕСКО
Е ВОЗДЕЙСТВИЕ F=A*COS(W*t+ω) ПР
И 0<t<=ТИ И F=0 ПРИ ТИ<t<ТК"
  70 PRINT ''"НАЖМИТЕ ЛЮБУЮ КЛАВ
ИШУ": PAUSE 0: CLS : PRINT "НАЧИ
НАЙТЕ ВВОД:"''"ЧИСЛО ВЕТВЕЙ N=";:
 INPUT "ВВЕДИТЕ ЧИСЛО ВЕТВЕЙ N="
;N: PRINT N
  80 DIM F(4): DIM G(50): DIM L(
50): DIM C(50): DIM U(25): DIM J
(25): DIM A(25): DIM B(25): DIM
 E(25): DIM D(25): DIM X(25): DIM
 Y(25)
  90 LET J1=1: LET J2=1: LET J3=
1
  95 PRINT TAB (0);"ТИП  ВЕТВЬ
НОМИНАЛ    ЕД.ИЗМЕР."
 100 INPUT "УКАЖИТЕ ТИП (G,L,C,F
,K,0)?  ";X$
 102 IF X$<>"G" AND X$<>"C" AND
X$<>"L" AND X$<>"F" AND X$<>"K"
AND X$<>"0" THEN GO TO 100
 105 IF X$="K" THEN GO TO 154
 110 IF X$="0" THEN GO TO 200
 120 IF X$="G" THEN INPUT "ВВЕДИ
ТЕ I,НОМ.G(1/ОМ)  ";I;"  ";X: PR
INT "G";"      ";I;"      ";X;TAB
(22);"1/ОМ": LET G(J1)=I: LET G(
J1+1)=X: LET J1=J1+2: GO TO 100
 130 IF X$="L" THEN INPUT "ВВЕДИ
ТЕ I,НОМ.L(ГН)  ";I;"  ";X: PRI
NT "L";"      ";I;"      ";X;TAB (
22);"ГН": LET L(J2)=I: LET L(J2+
1)=X: LET J2=J2+2: GO TO 100
 140 IF X$="C" THEN INPUT "ВВЕДИ
ТЕ I,НОМ.C(Ф)  ";I;"  ";X: PRINT
 "C";"      ";I;"      ";X;TAB (22
);"Ф": LET C(J3)=I: LET C(J3+1)=
X: LET J3=J3+2: GO TO 100
 150 IF X$="F" THEN INPUT "ВВЕДИ
ТЕ A(А ИЛИ В),W(УГЛ.),ω(РАД),ТИ
(С)  ";A;"  ";W;"  ";ω;"  ";T: GO T
O 165
 152 GO TO 100
 154 INPUT "УКАЖИТЕ ТИП КОРРЕКТИ
РУЕМОГО ЭЛЕМЕНТА  ";X$;"  ";X
 156 IF X$="G" THEN LET J1=(X+1)
/2: GO TO 95
 158 IF X$="L" THEN LET J2=(X+1)
/2: GO TO 95
 160 IF X$="F" THEN GO TO 95
 162 IF X$="C" THEN LET J3=(X+1)
/2: GO TO 95
 164 GO TO 154
 165 LET F$="A": IF N/2-INT (N/2
)<.1 THEN LET F$="B"
 170 PRINT '' ПАРАМЕТРЫ ИСТОЧНИК
А:"''A=";A;"  ";F$;"W=";W;"  РАД/С
","ω=";ω;"  РАД","ТИ=";T;"  С"
```

```
 180 LET F(1)=A: LET F(2)=W: LET
 F(3)=00: LET F(4)=T: GO TO 100
 200 PRINT "H=";: INPUT "ВВЕДИТЕ
 ШАГ H=";H: PRINT H
 210 PRINT "TK=";: INPUT "ВВЕДИТ
E ВРЕМЯ СЧЕТА TK=";T: PRINT T
 220 PRINT "ВВОД ЗАКОНЧЕН"'"НАЖМ
ИТЕ ЛЮБУЮ КЛАВИШУ": PAUSE 0
 230 CLS : PRINT "РЕЗУЛЬТАТЫ ВЫЧ
ИСЛЕНИЙ:"
 240 LET K4=T/H: DIM I(K4+1): DI
M V(N,K4+1)
 250 FOR J=1 TO 3: FOR I=1 TO 50
 STEP 2
 260 GO TO 260+20*J
 280 LET K=G(I): IF K=0 THEN GO
TO 350
 290 LET R=G(I+1): GO TO 340
 300 LET K=L(I): IF K=0 THEN GO
TO 350
 310 LET R=H/L(I+1): GO TO 340
 320 LET K=C(I): IF K=0 THEN GO
TO 350
 330 LET R=C(I+1)/H
 340 LET Y(K)=Y(K)+R: NEXT I
 350 NEXT J
 360 LET R=0: LET S=1
 370 FOR I=2 TO N STEP 2
 380 LET E(I-1)=Y(I-1)*S: LET R=
R+E(I-1)
 390 LET E(I)=R: LET A(I-1)=S: L
ET A(I)=R/Y(I)
 400 LET S=S+A(I): NEXT I
 410 LET M=N/2-INT (N/2): IF M<.
1 THEN GO TO 440
 420 LET E(N)=S*Y(N)
 430 LET A(N)=S: LET S=R+E(N)
 440 LET K1=0
 450 LET R=0: LET P=0
 460 FOR I=2 TO N STEP 2
 470 LET D(I-1)=Y(I-1)*P+X(I-1)
 480 LET R=R+D(I-1): LET D(I)=R
 490 LET B(I-1)=P: LET B(I)=(R-X
(I))/Y(I)
 500 LET P=P+B(I): NEXT I
 510 IF M<.1 THEN GO TO 540
 520 LET D(N)=P*Y(N)+X(N): LET B
(N)=P
 530 LET P=R+D(N)
 540 LET W=F(2): LET T1=K1*H
 550 IF W=0 THEN LET F1=F(1)
 560 IF T1>F(4) THEN LET F1=0
 570 IF F(4)<0 THEN LET F1=F(1)*
(1-EXP (T1/F(4)))
 580 IF W<>0 AND T1<F(4) THEN LE
T F1=F(1)*COS (W*T1+F(3))
 590 LET Z=(F1-P)/S: FOR I=1 TO
N
 600 LET U(I)=A(I)*Z+B(I)
 610 LET J(I)=E(I)*Z+D(I): NEXT
I
 620 LET K1=K1+1: LET T1=K1*H: I
F T1>T THEN GO TO 730
 630 PRINT "T=";T1,"F(T)=";F1: P
OKE 23692,255: FOR I=1 TO N
 640 PRINT "U";I;"=";U(I),"J";I;
"=";J(I): POKE 23692,255
 645 LET I(K1)=F1: LET V(I,K1)=U
(I)
 650 NEXT I
 660 FOR I=1 TO 50 STEP 2: LET K
=L(I)
 670 IF K=0 THEN GO TO 690
 680 LET X(K)=J(K): NEXT I
 690 FOR I=1 TO 50 STEP 2
 700 LET K=C(I): IF K=0 THEN GO
TO 720
 710 LET X(K)=-C(I+1)/H*U(K): NE
XT I
 720 GO TO 450
 730 PRINT "КОНЕЦ ВЫЧИСЛЕНИЙ"'"Е
СЛИ НУЖЕН ВЫВОД ГРАФИКОВ"'"НАЖМИ
ТЕ ЛЮБУЮ КЛАВИШУ"
```

```
 740 LET K=K1-1: PAUSE 0: CLS
1000 REM ВЫВОД ГРАФИКОВ
1010 CLS : GO SUB 9600: LET N4=2
00/K
1020 FOR X=0 TO 200-N4 STEP N4
1030 PLOT 30+X,10+75*I(1+X/N4)
1040 NEXT X
1050 FOR I=1 TO N: LET Y1=10: PL
OT 30,Y1
1055 IF I=2 OR I=6 OR I=7 THEN G
O TO 1095
1060 FOR X=0 TO 200-N4 STEP N4
1070 LET Y=10+75*U(I,1+X/N4)
1080 PLOT X+30,Y: DRAW N4,Y-Y1:
LET Y1=Y
1090 NEXT X
1095 NEXT I
1100 PRINT AT 0,5;"MY=2"
1110 PRINT AT 0,20;"MX=10 C"
1120 PRINT AT 17,21;"U8";AT 9,6;
"          U1   U3   U5";AT 21,9;"U
4"
1130 PAUSE 0: CLS : STOP
9600 REM ПОДПРОГРАММА ПОСТРОЕНИЯ
 КООРДИНАТНЫХ ОСЕЙ
9605 PLOT 30,160: DRAW 0,-150: D
RAW 200,0
9610 FOR O=50 TO 230 STEP 20
9615 PLOT O,10: DRAW 0,4: NEXT O
9620 FOR O=25 TO 160 STEP 15
9625 PLOT 30,O: DRAW 4,0: NEXT O
9630 PRINT AT 1,2;"1";AT 11,1;".
5";AT 21,2;"0"
9635 PRINT AT 21,15;".5": PRINT
AT 21,29;"1": RETURN
```

Контрольные примеры см. в § 7.12

Программа П1.9 (ZX-Spectrum)

Анализ нелинейных электронных цепей с топологическим вводом данных и обращением к встроенной расширяемой библиотеке элементов

```
  10 CLS : PRINT "АНАЛИЗ ЭЛЕКТРО
ННЫХ ЦЕПЕЙ""С ПОМОЩЬЮ ДИСКРЕТНЫ
Х СХЕМ""                ЗАМЕЩЕНИЯ"
  20 REM ВВОД И КОРРЕКЦИЯ ДАННЫХ
  30 PRINT ""ТОПОЛОГИЧЕСКИЕ МАСС
ИВЫ ЦЕПИ:""RI У1 У2 F NOM,R-МАС
СИВ R""CI У1 У2 F NOM,C-МАССИВ
C""LI У1 У2 F NOM,L-МАССИВ L"
  40 PRINT "II У1 У2 F NOM,I-МАС
СИВ I": LET J3=1: LET J4=1
  50 PRINT ""ЗАВИСИМЫЕ R,L,C,I З
АДАЮТСЯ УКАЗАНИЕМ F=-НС(НС)=2000
), ГДЕ НС-НОМЕР СТРОКИ ПОДПРОГРА
ММЫ,Y=NOM-ЗНАЧЕНИЕ R,L,C,I ИЛИ Н
ОМЕР УПРАВЛЯЮЩЕГО УЗЛА. УЗЛЫ ПОД
КЛЮЧЕНИЯ-К,К1"
  60 PRINT ""У1,У2-НОМЕРА УЗЛОВ"
""У ИСТОЧНИКОВ ТОКА ПЕРВЫМ УКАЗЫ
ВАЕТСЯ УЗЕЛ, ОТ КОТОРОГО ИДЕТ ТО
К,ПОСЛЕДНИЙ УЗЕЛ - ОБЩИЙ "
  70 LET Q$="НАЖМИТЕ ЛЮБУЮ КЛАВИ
ШУ": PRINT Q$: PAUSE 0: CLS
  80 PRINT ""F  ЗАДАЕТ СПЕЦИАЛЬН
УЮ ФУНКЦИЮ ЭЛЕМЕНТА: НОМЕР ПОДПР
ОГРАММЫ, ВИД ВОЗДЕЙСТВИЯ (ДЛЯ I)
 ИЛИ ДОПОЛНИТЕЛЬНЫЙ ПАРАМЕТР"
  90 PRINT ""ИСТОЧНИК ТОКА С F=0
 -ИМПУЛЬС""С  ЭКСПОНЕНЦИАЛЬНЫМИ
 ФРОНТАМИ""ВРЕМЕННЫЕ ПАРАМЕТРЫ=
ТЗ,ТИ,TF,TK""F>-2000 ЗАДАЕТ ПЬЕ
ДЕСТАЛ""""ПРИМЕР:""""I1  5  1  -
3000      .01""""ЗАДАННЫЙ ПОЛЬЗОВ
АТЕЛЕМ В ПОДПРОГРАММЕ С НАЧАЛОМ
В СТРОКЕ НС=3000""""ИСТОЧНИК ТОКА
 0.01 А, ВКЛЮЧЕННЫЙ""""МЕЖДУ УЗЛА
МИ К=5 И К1=1 И ВЫТЕКАЮЩИЙ ИЗ УЗ
ЛА 5"
```

```
 100 PRINT 'Q$: PAUSE 0: CLS
 110 INPUT "ИМЯ РАСЧЕТА? ";X$: P
RINT X$: LET J1=1: LET J2=1
 120 PRINT "ЧИСЛО УЗЛОВ N=";: IN
PUT "ЧИСЛО УЗЛОВ N=? ";N: PRINT
.N: LET N1=(2*N+1)*4: DIM J(N1)
 130 DIM Y(N): DIM R(N1): DIM C(
N1): DIM L(N1): LET N=N-1
 140 INPUT "ЗАДАЙТЕ L-ЕСЛИ R,C,L
 ЛИНЕЙНЫ И N-ЕСЛИ ОНИ НЕЛИНЕЙНЫ
";Z$: IF Z$="N" OR Z$="L" THEN G
O TO 160
 150 GO TO 140
 160 IF Z$="L" THEN PRINT "R,L,C
-ЛИНЕЙНЫ"
 170 IF Z$="N" THEN PRINT "R,L,C
-НЕЛИНЕЙНЫ"
 180 PRINT TAB (0);"ТИП  У1 У2 F
        НОМ."
 190 INPUT "ТИП (R,C,L,I,K,0) ?"
;X$
 200 IF X$<>"R" AND X$<>"C" AND
X$<>"L" AND X$<>"I" AND X$<>"K"
AND X$<>"0" THEN GO TO 190
 210 IF X$="0" THEN GO TO 380
 220 IF X$="K" THEN GO TO 310
 230 INPUT "У1 У2 F NOM. ? ";N2;
"  ";N3;"  ";N4;"  ";X
 240 IF X$="R" THEN PRINT TAB 0,
X$;(J1+3)/4;: LET R(J1)=N2: LET
R(J1+1)=N3: LET R(J1+2)=N4: LET
R(J1+3)=X: LET J1=J1+4
 250 IF X$="C" THEN PRINT TAB 0;
X$;(J2+3)/4;: LET C(J2)=N2: LET
C(J2+1)=N3: LET C(J2+2)=N4: LET
C(J2+3)=X: LET J2=J2+4
 260 IF X$="L" THEN PRINT TAB 0;
X$;(J3+3)/4;: LET L(J3)=N2: LET
L(J3+1)=N3: LET L(J3+2)=N4: LET
L(J3+3)=X: LET J3=J3+4
 270 IF X$="I" THEN PRINT TAB 0,
X$;(J4+3)/4;: LET J(J4)=N2: LET
J(J4+1)=N3: LET J(J4+2)=N4: LET
J(J4+3)=X: LET J4=J4+4
 280 PRINT TAB 5;N2;TAB 8;N3;: I
F N4<=-2000 THEN PRINT TAB 11;"П
П*";-N4.
 290 IF N4>-2000 THEN PRINT TAB
11;N4;
 300 PRINT TAB 20;X: GO TO 190
 310 INPUT "УКАЖИТЕ КОРРЕКТИРУЕМ
ЫЙ ЭЛЕМЕНТ? ";X$: LET X=4*VAL X$
(2 TO )-3: LET X$=X$(1)
 320 IF X$="R" THEN LET J1=X
 330 IF X$="C" THEN LET J2=X
 340 IF X$="L" THEN LET J3=X
 350 IF X$="I" THEN LET J4=X
 360 IF X$="R" OR X$="C" OR X$="
L" OR X$="I" THEN GO TO 180
 370 GO TO 310
 380 PRINT TAB (0);"ШАГ H=";: IN
PUT "ЗАДАЙТЕ ШАГ H=";H: PRINT H,
 390 INPUT "ВВЕДИТЕ ВРЕМЕНА T3,T
И,TФ,TK ";T3;" ";TI;" ";TF;" ";T
K: PRINT "ВРЕМЕНА ИМПУЛЬСА"
 400 PRINT "T3=";T3,"TИ=";TI,"TФ
=";TF,"TK=";TK
 410 LET K5=TK/H: DIM Y(N+1): DI
M Q(N+1): DIM P(N
 420 INPUT "НОМЕР ГРАФИКА ВОЗДЕЙ
СТВИЯ(ИЛИ 0)";GI: IF GI<>0 THEN
INPUT "МАСШТАБНОЕ СОПРОТИВЛЕНИЕ
? ";RI: DIM I(K5)
 430 INPUT "НОМЕРА ТРЕХ ВРЕМЕННЫ
Х ЗАВИСИМОСТЕЙ (ИЛИ НУЛИ) ? ";G1
;" ";G2;" ";G3. IF G1<>0 THEN DI
M V(K5)
 440 IF G2<>0 THEN DIM U(K5)
 450 IF G3<>0 THEN DIM W(K5)
 460 PRINT "ВВЕДИТЕ ДАННЫЕ ПОДП
РОГРАММ","И ДАЙТЕ КОМАНДУ CONTIN
UE": STOP : CLS : LET K4=1
```

```
 470 REM ФОРМИРОВАНИЕ МАТРИЦЫ УЗ
ЛОВЫХ ПРОВОДИМОСТЕЙ Z(N,N)
 480 LET T2=T3+TI: DIM Z(N+1,N+1
): DIM X(N+1): DIM M(N)
 490 FOR I=1 TO N1 STEP 4: LET O
=I+3: LET O2=I+2: LET O1=I+1: IF
 R(I)=0 AND C(I)=0 AND L(I)=0 TH
EN GO TO 630
 500 FOR J=1 TO 3: IF J=2 THEN G
O TO 550
 510 IF J=3 THEN GO TO 580
 520 LET K=R(I): LET K1=R(O1): I
F R(O2)<0 THEN LET Y=R(O): GO SU
B -R(O2): GO TO 540
 530 LET R=R(O): IF R=0 THEN GO
TO 620
 540 LET P(I)=R: LET I1=1/R: GO
TO 610
 550 LET K=L(I): LET K1=L(O1): I
F L(O2)<0 THEN LET Y=L(O): GO SU
B -L(O2): GO TO 570
 560 LET L=L(O): IF L=0 THEN GO
TO 620
 570 LET P(O1)=L: LET I1=H/L: GO
TO 610
 580 LET K=C(I): LET K1=C(O1): I
F C(O2)<0 THEN LET Y=C(O): GO SU
B -C(O2): GO TO 600
 590 LET C=C(O): IF C=0 THEN GO
TO 620
 600 LET P(O2)=C: LET I1=C/H
 610 LET Z(K,K)=Z(K,K)+I1: LET Z
(K1,K1)=Z(K1,K1)+I1: LET Z(K,K1)
=Z(K,K1)-I1: LET Z(K1,K)=Z(K1,K)
-I1
 620 NEXT J: NEXT I
 630 REM ОБРАЩЕНИЕ К ПОДПРОГРАММ
Е ОБРАЩЕНИЯ МАТРИЦЫ Z(N,N)
 640 GO SUB 930
 650 REM ФОРМИРОВАНИЕ ВЕКТОРА ДЕ
ЙСТВУЮЩИХ ТОКОВ
 660 LET T=K4*H: FOR I=1 TO 100:
 LET J=I*4: IF J(J-3)=0 THEN GO
TO 740
 670 LET K=J(J-3): LET K1=J(J-2)
  LET K3=J(J-1): IF K3<=-2000 TH
EN LET Y=J(J): GO SUB -K3: GO TO
 720
 680 IF T>T3 AND T<=T2 THEN LET
I1=J(J)*(1-EXP (-(T-T3)/TF)): LE
T P(J)=I1
 690 IF T>T2 THEN LET I1=P(J)*EX
P (-(T-T2)/TF): GO TO 710
 700 IF T<=T3 THEN LET I1=0
 710 IF K3>-2000 THEN LET I1=I1+
K3
 720 LET X(K)=X(K)-I1: LET X(K1)
=X(K1)+I1: IF I=GI AND GI<>0 THE
N LET I(K4)=I1*RI
 730 NEXT I
 740 FOR I=1 TO 200 STEP 4: LET
O1=I+1: LET O2=I+2: LET O3=I+3:
IF L(I)=0 AND C(I)=0 THEN GO TO
810
 750 LET K=L(I): LET K1=L(O1): L
ET J=C(I): LET K3=C(O1): IF K=0
THEN GO TO 780
 760 LET I1=(Y(K)-Y(K1))*H/P(O1)
+Q(I)
 770 LET I1=(Y(K)-Y(K1))*H/P(O1)
+Q(I): LET Q(I)=I1: LET X(K)=X(K
)-I1: LET X(K1)=X(K1)+I1
 780 IF J=0 THEN GO TO 800
 790 LET I1=(Y(J)-Y(K3))*P(O2)/H
: LET X(J)=X(J)+I1: LET X(K3)=X(
K3)-I1
 800 NEXT I
 810 PRINT "T=";T
 820 REM ВЫЧИСЛЕНИЕ ВЕКТОРА УЗЛО
ВЫХ ПОТЕНЦИАЛОВ
 830 FOR I=1 TO N: LET Y(I)=0: F
OR J=1 TO N: LET Y(I)=Y(I)+Z(I,J
)*X(J): NEXT J: NEXT I
```

```
 840 FOR I=1 TO N: LET X(I)=0: I
F I=G1 AND G1<>0 THEN LET U(K4)=
Y(I)
 850 IF I=G2 AND G2<>0 THEN LET
U(K4)=Y(I)
 860 IF I=G3 AND G3<>0 THEN LET
U(K4)=Y(I)
 870 PRINT "U";I;"=";Y(I): POKE
23692,255
 880 NEXT I: LET K4=K4+1: IF K4-
1<=K5 THEN GO TO 810
 890 IF Z$="N" THEN GO TO 470
 900 GO TO 660
 910 PRINT "КОНЕЦ ВЫЧИСЛЕНИЙ"'"Е
СЛИ НУЖЕН ВЫВОД ГРАФИКОВ"'"НАЖМИ
ТЕ ЛЮБУЮ КЛАВИШУ"
 920 PAUSE 0: GO TO 1060
 930 REM ПОДПРОГРАММА ОБРАЩЕНИЯ
МАТРИЦЫ ПРОВОДИМОСТЕЙ Z(N,N)
 940 LET D=1: LET R=1: FOR I=1 T
O N: FOR J=I TO N: IF Z(J,I)<>0
THEN GO TO 960
 950 NEXT J: PRINT "МАТРИЦА ВЫРО
ЖДЕНА": STOP
 960 FOR K=1 TO N: LET B=Z(I,K):
 LET Z(I,K)=Z(J,K): LET Z(J,K)=B
: NEXT K: LET M(I)=J: LET Z(I,I)
=1/Z(I,I)
 970 FOR K=1 TO N: IF K=I THEN G
O TO 990
 980 LET Z(I,K)=Z(I,I)*Z(I,K): N
EXT K: GO TO 1000
 990 NEXT K
1000 FOR J=1 TO N: IF J=I THEN G
O TO 1030
1010 LET B=-Z(J,I): LET Z(J,I)=0
1020 FOR K=1 TO N: LET Z(J,K)=Z(
J,K)+B*Z(I,K): NEXT K
1030 NEXT J: LET D=D/Z(I,I): NEX
T I: FOR I=N TO 1 STEP -1: IF M(
I)=I THEN GO TO 1050
1040 LET R=-R: FOR K=1 TO N: LET
 B=Z(K,I): LET Z(K,I)=Z(K,M(I)):
 LET Z(K,M(I))=B: NEXT K
1050 NEXT I: RETURN
1060 REM ВЫВОД ГРАФИКОВ
1070 CLS : GO SUB 1310: LET N4=2
20/(K4-1)
1080 LET X=35: LET Y=15+Y4*16: P
LOT X,Y
1090 FOR J=0 TO N: LET X=35: LET
 Y=15+Y4*16: LET Y2=Y: PLOT X,Y
1100 INPUT "ЧЕМ СТРОИТЬ (PLOT/DR
AW)? ";G$
1110 FOR I=1 TO K4-1
1120 IF J=0 AND G1<>0 THEN LET Y
1=Y2+Y5*I(I)
1130 IF J=G1 AND G1<>0 THEN LET
Y1=Y2+Y5*U(I)
1140 IF J=G2 AND G2<>0 THEN LET
Y1=Y2+Y5*U(I)
1150 IF J=G3 AND G3<>0 THEN LET
Y1=Y2+Y5*U(I)
1160 LET X1=35+I*N4: IF G$="D" T
HEN DRAW X1-X,Y1-Y
1170 LET Y=Y1: LET X=X1: PLOT X,
Y
1180 NEXT I: NEXT J
1190 INPUT "ВЫДАТЬ КОММЕНТАРИИ (Y
/N) ";K$: IF K$="Y" THEN GO TO 1
210
1200 PAUSE 0: GO TO 1060
1210 INPUT "КУРСОР-ENTER, ТЕКСТ-
T";Z$
1220 LET X=15: LET Y=10
1230 LET A$=INKEY$: BEEP .05,20
1240 IF A$=CHR$ 8 THEN LET X=X-1
: IF X<0 THEN LET X=0
1250 IF A$=CHR$ 10 THEN LET Y=Y+
1: IF Y<0 THEN LET Y=0
1260 IF A$=CHR$ 11 THEN LET Y=Y-
1: IF Y>21 THEN LET Y=21
```

```
1270 IF A$=CHR$ 9 THEN LET X=X+1
: IF X>31 THEN LET X=31
1280 PRINT OVER 1; PAPER 3;AT Y,
X;" "
1290 IF A$>CHR$ 32 AND A$<CHR$ 1
65 THEN PRINT OVER 1; PAPER 7;AT
 Y,X;" ": INPUT "КОМ.?";A$: PRIN
T OVER 1;AT Y,X;A$: INPUT "КОММЕ
НТАРИЙ ВЕРЕН (Y/N)? ";B$: IF B$=
"N" THEN PRINT OVER 1;AT Y,X;A$
1300 PAUSE 0: PRINT OVER 1; PAPE
R 7;AT Y,X;" ": GO TO 1230
1310 REM ПОДПРОГРАММА ОЦИФРОВКИ
И ПОСТРОЕНИЯ КООРДИНАТНЫХ ОСЕЙ
1320 IF TM<=1E-6 THEN LET T$=" Н
С": LET TM=TM/1E-9
1330 IF TM>1E-6 AND TM<=1E-3 THE
N LET T$="МКС": LET TM=TM/1E-6
1340 IF TM>1E-3 AND TM<=1 THEN L
ET T$=" МС": LET TM=TM/1E-3
1350 IF TM>=1 THEN LET T$="  С":
 LET TM=TM
1360 PRINT "МАСШТАБ ПО ОСИ U (КВ
/В/МВ)? ";: INPUT V$
1370 PRINT V$: PRINT "ЗАДАЙТЕ UM
IN=";: INPUT "UMIN=";U8: PRINT U
8
1380 PRINT "ЗАДАЙТЕ UMAX=";: INP
UT "UMAX=";U9: PRINT U9
1390 LET Y4=10*U8/(U8-U9): LET Y
5=160/(U9-U8): IF V$="MB" THEN L
ET Y5=1E3*Y5
1400 LET X5=220/TM
1410 CLS : IF U8<=0 AND U9>=0 TH
EN PRINT AT 20-2*Y4,3;0
1420 LET X=LEN (STR$ U9): IF X>4
 THEN LET X=4
1430 PRINT AT 0,4-X;U9: PRINT AT
 1,3;"В": IF V$="MB" OR V$="KB"
THEN PRINT AT 1,2;V$
1440 LET X=LEN (STR$ ((U8+U9)/2)
): IF X>4 THEN LET X=4
1450 PRINT AT 10,4-X;(U8+U9)/2
1460 LET X=LEN (STR$ U8): IF X>4
 THEN LET X=4: IF U8<>0 THEN PRI
NT AT 19,4-X;U8
1470 IF U8<>0 THEN PRINT AT 19,4
-X;U8
1480 LET X=LEN (STR$ TM): PRINT
AT 20,17;TM/2;TAB 32-X;TM: PRINT
 AT 21,29;T$
1490 REM ПОСТРОЕНИЕ КООРДИНАТНОЙ
 СИСТЕМЫ
1500 PLOT 35,175: DRAW 0,-160: D
RAW 220,0: DRAW 0,160: DRAW -220
,0
1510 FOR I=0 TO 9
1520 PLOT 35,16*I+15: DRAW 3,0:
PLOT 35,16*I+23: DRAW 1,0
1530 PLOT 255,16*I+15: DRAW -3,0
: PLOT 255,16*I+23: DRAW -1,0
1540 PLOT 35+22*I,175: DRAW 0,-3
: IF I=5 THEN DRAW 0,-1
1550 PLOT 46+22*I,175: DRAW 0,-1
:
1560 PLOT 35+22*I,15: DRAW 0,3:
IF I=5 THEN DRAW 0,2
1570 PLOT 46+22*I,15: DRAW 0,1:
NEXT I
1580 IF U8<0 AND U9>0 THEN FOR I
=0 TO 19: PLOT 35+11*I,16*Y4+15:
 NEXT I
1590 RETURN
2000 REM ПОДПРОГРАММЫ ЗАДАНИЯ ЗА
ВИСИМЫХ R,L,C,I
2010 REM ЗАДАНИЕ ИМПУЛЬСА С ЛИНЕ
ЙНЫМ НАРАСТАНИЕМ И СПАДОМ
2020 IF T>=0 AND T<=TF THEN LET
I1=Y*T/TF: LET IM=I1: RETURN
2030 LET I1=IM: IF T>TI THEN LET
 I1=IM-Y*(T-TI)/TF
2040 IF I1<0 THEN LET I1=0
2050 RETURN
```

```
2100 REM ЗАДАНИЕ СИНУСОИДАЛЬНОГО
ВОЗДЕЙСТВИЯ
2110 LET I1=Y+Y*SIN (2*PI*T/TI+T
F): RETURN
3000 REM ВАХ GaAs ТУННЕЛЬНОГО ДИ
ОДА
3010 LET I1=J(J)*27.18*Y(2)*EXP
(-10*Y(2))+1E-8*(EXP (20*Y(2))-1
): RETURN
4000 REM НЕЛИНЕЙНАЯ ЕМКОСТЬ
4010 LET C=(10E-12)*50R (1.2/(1.
2-Y(Y)))+5E-12: RETURN
5000 REM МОДЕЛЬ Si ДИОДА
5010 LET I1=Y*(EXP ((Y(K)-Y(K1))
*20)-1): RETURN
5020 GO SUB 5010: LET C=5E-12*50
R (1/(1-Y(K)+Y(K1)))+I1*20*Y: RE
TURN
6000 REM МОДЕЛЬ Si  БИПОЛЯРНОГО
ТРАНЗИСТОРА
6010 GO SUB 6050: LET I1=IN*(1+1
/Y)-II: RETURN
6020 GO SUB 6060: LET I1=II*(1+1
/Y)-IN: RETURN
6030 GO SUB 6050: LET C=Y*50R (1
/(1-Y(K)+Y(K1)))+3E-8*IN: RETURN

6040 GO SUB 6060: LET C=Y*50R (1
/(1-Y(K)+Y(K1)))+30E-8*II: RETUR
N
6050 LET IN=.2E-9*(EXP (30*(Y(K)
-Y(K1))-1)): RETURN
6060 LET II=1E-9*(EXP (30*(Y(K)-
Y(K1))-1)): RETURN
7000 LET I1=Y: RETURN
8000 LET I1=Y: RETURN
9999 LOAD "RALF"CODE : RUN 10
```

Контрольные примеры см. в § 7.12.

Программа П1.10 (ZX-Spectrum)

Спектральный анализ таблично заданных сигналов с выводом графиков

```
  10 PRINT "СПЕКТРАЛЬНЫЙ АНАЛИЗ"
  PRINT
  20 PRINT "ВВЕДИТЕ ЧИСЛО ИНТЕРВ
АЛОВ РАЗБИЕНИЯ N=";: INPUT N: PR
INT N
  30 PRINT "ВВЕДИТЕ НОМЕР ПЕРВОГ
О";: PRINT "НЕНУЛЕВОГО ОТСЧЕТА N1
=";: INPUT NO: PRINT NO
  40 PRINT "ВВЕДИТЕ НОМЕР ПОСЛЕД
НЕГО НЕНУЛЕВОГО ОТСЧЕТА M=";: IN
PUT M: PRINT M: DIM Y(N+1)
  45 FOR I=0 TO NO: LET Y(I+1)=0
: NEXT I
  50 FOR I=M+2 TO N+1: LET Y(I)=
0: NEXT I
  55 LET G=0: FOR I=NO TO M
  60 PRINT "ВВЕДИТЕ ОТСЧЕТ Y";I;
"=";: INPUT Y(I+1)
  65 IF G<Y(I+1) THEN LET G=Y(I+
1)
  70 PRINT Y(I+1): NEXT I
  80 PRINT "ЗАДАЙТЕ ОБЩИЙ ИНТЕРВ
АЛ T0=";: INPUT TO: PRINT TO: LE
T T=TO/N
  85 PRINT "ВВЕДИТЕ ЧИСЛО ИНТЕРВ
АЛОВ СПЕКТРА K=";: INPUT K: PRIN
T K
  90 DIM S(K+1): DIM C(K+1): DIM
 A(K+1): DIM Q(K+1)
  95 PRINT "ЗАДАЙТЕ МАКСИМАЛЬНУЮ
 ЧАСТОТУ СПЕКТРА F МАКС.=";: INP
UT FO: LET FT=FO/K: PRINT FO
 100 FOR J=1 TO K+1: LET C=0: LE
T S=0: LET F=(J-1)*FT
```

```
 110 LET P=PI*F*T: LET W=2*P
 120 FOR I=N0 TO M
 130 LET Z=W*I: LET C=C+Y(I+1)*C
OS (Z)
 140 LET S=S+Y(I+1)*SIN (Z): NEX
T I
 145 LET S(J)=S: LET C(J)=C
 150 LET R=SQR (C*C+S*S): LET Q=
0: IF R>1E-5 THEN LET Q=-ACS (C/
R-C*1E-6)
 155 IF S<0 THEN LET Q=-Q
 160 LET Q=Q-PI*T*F
 170 LET K0=1: IF P<>0 THEN LET
K0=SIN (P)/P
 180 LET R0=K0*R: LET RT=K0*R0
 185 LET A(J)=RT*T: LET Q(J)=Q:
IF A(J)<1E-30 THEN LET Q(J)=0
 190 NEXT J
 200 CLS : PRINT AT 5,0;"ЗАДАЙТЕ
 КОД;": PRINT "0-ВЫВОД ОТСЧЕТОВ
Y(I)": PRINT "1-ПОСТРОЕНИЕ ГРАФИ
КА Y(T)"
 210 PRINT "2-ВЫВОД ТАБЛИЦЫ C(I)
 И S(I)": PRINT "3-ВЫВОД ТАБЛИЦЫ
 A(F) И Q(F)"
 220 PRINT "4-ВЫВОД ГРАФИКА A(F)
": PRINT "5-ВЫВОД ГРАФИКА Q(F)"
 INPUT X$
 230 IF X$="0" THEN GO TO 280
 240 IF X$="1" THEN GO TO 310
 250 IF X$="2" THEN GO TO 290
 260 IF X$="3" THEN GO TO 300
 270 IF X$="4" THEN GO TO 370
 275 IF X$="5" THEN GO TO 440
 280 CLS : PRINT "ОТСЧЕТЫ Y(I)":
 PRINT : FOR I=0 TO N: PRINT "Y"
;I;"=";Y(I+1): NEXT I: PAUSE 0:
GO TO 200
 290 CLS : PRINT "C(F) И S(F)":
PRINT : FOR I=0 TO K: PRINT "F="
;I*FT;TAB 12;"C(F)=";C(I+1): PRI
NT TAB 12;"S(F)=";S(I+1): NEXT I
: PAUSE 0: GO TO 200
 300 CLS : PRINT "A(F) И Q(F)":
PRINT : FOR I=0 TO K: PRINT "F="
;I*FT: PRINT "A(F)=";A(I+1): PRI
NT "Q(F)=";Q(I+1);" РАД": PRINT
"Q(F)=";Q(I+1)*180/PI;" ГРАД": N
EXT I: PAUSE 0: GO TO 200
 310 CLS : LET X0=50: LET Y0=95+
75*Y(1)/G: PLOT X0,Y0
 320 FOR I=1 TO N: LET YT=95+75*
Y(I+1)/G
 330 DRAW 200/N,YT-Y0: LET Y0=YT
: NEXT I
 340 GO SUB 800: PRINT AT 12,28;
"T/T0"
 350 PRINT AT 1,1;"Y/YM": PRINT
AT 21,1;"YM=";G
 360 PRINT AT 21,20;"T0=";T0: PA
USE 0: GO TO 200
 370 CLS : LET H=200/K: LET X0=5
0: LET Y0=20+100*A(1)/A(2)
 380 FOR I=1 TO K: LET YT=20+100
*A(I+1)/A(2)
 390 PLOT X0,Y0: DRAW H,YT-Y0: L
ET Y0=YT: LET X0=X0+H: NEXT I
 400 FOR I=1 TO K: PLOT 50+I*H,2
0: LET Y=100*A(I+1)/A(2): DRAW 0
,Y: NEXT I
 410 GO SUB 900: PRINT AT 1,1;"A
/A1": PRINT AT 21,28;"F/FM"
 420 PRINT AT 21,0;"A1=";A(2): P
RINT AT 21,16;"FM=";FQ
 430 PAUSE 0: GO TO 200
 440 CLS : LET Y0=95: FOR I=1 TO
K: PLOT 50+I*200/K,95
 450 LET Y=75*Q(I)/PI: DRAW 0,Y
 460 NEXT I
 470 GO SUB 800: PRINT AT 1,1;"Q
/PI"
 480 PRINT AT 12,28;"T/T0": PRIN
T AT 21,15;"FM=";FQ
```

```
  490 PAUSE 0: GO TO 200
  800 PLOT 50,20: DRAW 0,150
  810 PLOT 50,95: DRAW 200,0
  820 FOR X=50 TO 250 STEP 20
  830 PLOT X,95: DRAW 0,-4: NEXT
X
  840 FOR Y=20 TO 170 STEP 15
  850 PLOT 50,Y: DRAW -4,0: NEXT
Y
  860 PRINT AT 19,4;"-1": PRINT A
T 10,5;"0": PRINT AT 0,4;"+1"
  870 PRINT AT 11,18;".5": PRINT
AT 11,31;"1": RETURN
  900 PLOT 50,20: DRAW 200,0
  910 PLOT 50,20: DRAW 0,150
  920 FOR X=50 TO 250 STEP 20
  930 PLOT X,20: DRAW 0,-4: NEXT
X
  940 FOR Y=20 TO 170 STEP 10
  950 PLOT 50,Y: DRAW -4,0: NEXT
Y
  960 PRINT AT 20,5;"0": PRINT AT
  13,4;".5": PRINT AT 6,5;"1"
  970 PRINT AT 0,3;"1.5": PRINT A
T 20,18;".5"
  980 PRINT AT 20,31;"1": RETURN
```

Контрольные примеры см. в § 7.13

Программа П1.11 (Д3-28)

Прямое и обратное преобразование Фурье

```
10 PRINT'ПРЯМОЕ И ОБРАТНОЕ БЫСТРОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ'
20 INPUT'ДЛЯ N=2^M ВВЕДИТЕ M='M:LETN=INT(2^M+.1):DIM X(N-1),Y(N-1)
30 INPUT'ЗАДАЙТЕ -1 ПРИ ПРЯМОМ БПФ И 1 ПРИ ОБРАТНОМ 'D
35 INPUT'ВВЕДИТЕ НОМЕРА НАЧАЛЬНОГО И КОНЕЧНОГО ОТСЧЕТОВ IS,IF 'I0,I1
40 FOR I=0 TO I1-I0:PRINT!3.0!'ДЛЯ I='I+I0
50 INPUT'ВВЕДИТЕ   XI,YI   'X(I),Y(I):NEXT I
60 FOR L=1 TO M:LETE=INT(2^(M+1-L)+.1):LETF=E/2:LETU=1:LETV=0
70 LETZ=#PI/F:LETC=COS(Z):LETS=D*SIN(Z):FOR J=1 TO F
80 FOR I=J TO N STEP E:LET0=I+F-1:LETP=X(I-1)+X(0):LET0=Y(I-1)+Y(0)
90 LETR=X(I-1)-X(0):LETT=Y(I-1)-Y(0):LETX(0)=P*U-T*V
100 LETY(0)=T*U+R*V:LETX(I-1)=P:LETY(I-1)=0:NEXT I
110 LETW=U*C-V*S:LETV=V*C+U*S:LETU=W:NEXT J:NEXT L
120 LETJ=1:FOR I=1 TO N-1:IF I>=J THEN 150
130 LETJ1=J-1:LETI1=I-1:LETP=X(J1):LET0=Y(J1):LETX(J1)=X(I1)
140 LETY(J1)=Y(I1):LETX(I1)=P:LETY(I1)=0
150 LETK=N/2
160 IF K>=J THEN 180
170 LETJ=J-K:LETK=K/2:GOTO 160
180 LETJ=J+K:NEXT I
190 IF D=-1 THEN PRINT'РЕЗУЛЬТАТЫ ПРЯМОГО БПФ':GOTO 240
200 PRINT'РЕЗУЛЬТАТЫ ОБРАТНОГО БПФ':FOR K=0 TO N-1
210 LETX(K)=X(K)/N:LETY(K)=Y(K)/N
220 PRINT!3.0!'K='K;!F1.5!'  X='X(K);'  Y='Y(K)
230 NEXT K:STOP
240 FOR K=0 TO N-1:LETA=SQR(X(K)*X(K)+Y(K)*Y(K))
250 LET0=0:IF A=0 THEN 270
260 LET0=ACS(X(K)/A):IF Y(K)<0 THEN LET0=-0
270 PRINT !3.0!'K='K;!F1.5!'  X='X(K);'  Y='Y(K);
280 PRINT'  M='A*2/N;'  0='DEG(0):NEXT K:END
```

Контрольный пример см. в § 7.13.

Графический редактор

```
  10 REM GRAFU
  15 GO TO 9000
  20 FOR i=1 TO n: LET i1=ABS y(
i)
  30 LET j=INT ((i1-INT i1)*10+.
01): INK j
  40 LET p1=ABS x(i): LET p=INT
((p1-INT p1)*10+.01)
  50 LET x=(SGN x(i))*p1: LET y=
(SGN y(i))*INT (ABS y(i))
  70 IF p=0 THEN PLOT x,y
  80 IF p=1 THEN DRAW x,y,r(i)
  90 IF p=2 THEN CIRCLE x,y,r(i)
 100 IF p=3 THEN PRINT AT y,x;c$
(i)
 110 NEXT i: STOP : GO TO 20
9000 PRINT : PRINT : PRINT : PRI
NT : PRINT "        ***GRAFIC SAVE
***"
9005 PRINT : PRINT : PRINT : PRI
NT " 1.GRAFIC REDACTOR": PRINT
" 2.GRAFIC PROGRAMM": PRINT " 3.
GRAFIC SAVE": PRINT "  ARRAYS: x
(i),y(i),r(i),c$(i)"
9010 PAUSE 100: CLS : INPUT "BOR
DER =";b
9020 INPUT "PAPER=";c: BORDER b:
 PAPER c
9030 INPUT "DIM=";n1: DIM x(n1):
 DIM y(n1): DIM r(n1): DIM c$(n1
): LET n=0
9040 INK 0: PRINT n;" BORDER ";b
;": PAPER ";c: INPUT "INK=";j
9050 LET x=10: LET x1=x: LET y=9
0: INK 0: PLOT x,y: PRINT AT 12,
1;"↑ Marker": PAUSE 200: CLS
9055 LET n=1: LET k=1: LET j=0:
LET l=0: CLS
9060 IF INKEY$="6" THEN FOR y=y
TO y STEP -1: NEXT y: BEEP .001,
20
9070 IF INKEY$="7" THEN FOR y=y
TO y: NEXT y: BEEP .001,20
9080 IF INKEY$="8" THEN FOR x=x
TO x: NEXT x: BEEP .001,20
9090 IF INKEY$="5" THEN FOR x=x
TO x STEP -1: NEXT x: BEEP .001,
20
9100 PLOT OVER 0;x,y: PLOT OVER
0;x,y: PLOT OVER 0;x,y: PLOT OVE
R 0;x,y
9110 PLOT OVER 1;x,y: INK j
9115 IF INKEY$="0" THEN LET k=1:
 GO TO 9170
9120 IF INKEY$="1" THEN LET k=0:
 GO TO 9170
9125 IF k=0 THEN PLOT OVER 1;x,y
9130 IF INKEY$="i" THEN GO TO 91
80
9145 IF INKEY$="l" THEN LET r=0:
 LET z$="l": GO TO 9205
9150 IF INKEY$="q" THEN LET z$="
q": GO TO 9205
9160 IF INKEY$=" " THEN GO TO 94
50
9170 GO TO 9060
9180 INPUT "INK x,DRAW w,CIRCLE
h,PIXEL p =";z$
9200 IF z$="x" THEN LET j1=j: IN
PUT "Ink c=";j: GO TO 9060
9205 INK c: PRINT AT 0,0;"      ": INK
 j
9208 INK 0: OVER 0: PRINT AT 0,0
;n;" INK ";j;": ";: INK j
9210 IF z$="q" THEN GO TO 9260
9220 IF z$="w" THEN INPUT "r=";r
: GO TO 9300
```

```
9225 IF z$="l" THEN GO TO 9300
9230 IF z$="h" THEN GO TO 9370
9240 IF z$="p" THEN GO TO 9410
9250 GO TO 9160
9260 BEEP .001,20: LET l=0: PLOT
 x,y: LET x1=x: LET y1=y: LET x=
x+1: LET n=n+1
9270 LET x(n)=x1+.1*l: LET y(n)=
y1+.1*j: LET r(n)=0
9290 INK 0: PRINT " PLOT ";x1;",
";y: INK j: GO TO 9448
9300 BEEP .001,20: LET l=1: LET
x2=x-x1: LET y2=y-y1: LET n=n+1:
 PLOT x1,y1
9310 DRAW x2,y2,r: LET x4=x1: LE
T y4=y1: LET x1=x: LET y1=y
9320 LET x(n)=x2+.1*l*SGN x2: LE
T y(n)=y2+.1*j*SGN y2: LET r(n)=
r
9330 IF x2=0 THEN LET x(n)=.1*l
9335 IF y2=0 THEN LET y(n)=.1*j
9340 INK 0: PRINT " DRAW ";x2;"
,";y2;
9350 IF r<>0 THEN PRINT ",";r
9360 INK j: GO TO 9448
9370 LET l=2: INPUT "Radius r=";
r1: CIRCLE x,y,r1: LET n=n+1
9380 LET x(n)=x+.1*l: LET y(n)=y
+.1*j: LET r(n)=r1
9400 INK 0: PRINT " CIRCLE ";x;"
,";y;",";r1: INK j: GO TO 9448
9410 LET n=n+1: LET l=3: LET x3=
INT (x/8)
9420 LET y3=21-INT (y/8): PRINT
AT y3,x3;"■"
9430 PAUSE 30: OVER 1: PRINT AT
y3,x3;"■"
9435 INPUT p$: INK 0: PRINT AT y
3,x3;p$: INK j
9440 LET y(n)=y3+.1*j: FOR i=0 T
O LEN p$-1
9442 LET c$(n+i)=p$(i+1 TO i+1):
 LET y(n+i)=y3: LET x(n+i)=i+x3+
.1*l: NEXT i: LET n=n-1+LEN p$
9445 OVER 0: PRINT AT 0,0;n;" IN
K ";j;";" PRINT AT ";y3;",";x3
;",";p$;"■"
9446 BEEP 1,5: GO TO 9060
9450 IF l=0 THEN LET x=x-1: PLOT
 OVER 1;x,y
9460 IF l=1 THEN PLOT x4,y4: DRA
W OVER 1;x2,y2,r: PLOT OVER 1;x,
y: LET x=x4: LET y=y4: LET x1=x:
 LET y1=y: LET x=x+1
9470 IF l=2 THEN CIRCLE OVER 1;x
,y,r1
9480 IF l=3 THEN PRINT OVER 1;AT
 y3,x3;p$: IF LEN p$>1 THEN LET
n=n+1-LEN p$
9490 PAUSE 50: LET n=n-1: INK c
9500 PRINT AT 0,0;"                ": INK j
9510 BEEP 1,5: GO TO 9150
```

ПРИЛОЖЕНИЕ 2

ПАКЕТ ДЕМОНСТРАЦИОННЫХ ПРОГРАММ ДЛЯ ПЕРСОНАЛЬНЫХ ЭВМ IBM PC

Версия Бейсика IBM PC является одной из наиболее распространенных [53, 57] и используется в ряде отечественных ПЭВМ (ЕС-1842, ЕС-1841, "Искра-1030" и др.). Приведенные ниже программы иллюстрируют возможности этой версии, описанные в основных разделах книги.

Программа П2.1

Точное вычисление таблицы факториалов $N!$

```
10 PRINT "PRINTS EXACT TABLE OF FACTORIALS"
20 DIM A(200):W=52:INPUT "INPUT MAX FACTORIAL";N
30 A(1)=1:PRINT "NUMBER     FACTORIAL  "
40 FOR I=1 TO N:B=I
50   FOR J=1 TO 200
60     A(J)=A(J)*B+C:C=INT(A(J)/10):A(J)=A(J)-10*C
70       IF C=0 AND J>=S GOTO 90
80   NEXT J
90   S=J
100  PRINT I TAB(11);
110  FOR L=S TO 1 STEP -1
120  PRINT MID$(STR$(A(L)),2);:SL=S-L+2
130    IF SL=W*INT(SL/W) THEN PRINT  CHR$(10) TAB(11);
140   NEXT L
150   PRINT
160 NEXT I

RUN
PRINTS EXACT TABLE OF FACTORIALS
INPUT MAX FACTORIAL? 18
NUMBER     FACTORIAL
 1         1
 2         2
 3         6
 4         24
 5         120
 6         720
 7         5040
 8         40320
 9         362880
 10        3628800
 11        39916800
 12        479001600
 13        6227020800
 14        87178291200
 15        1307674368000
 16        20922789888000
 17        355687428096000
 18        6402373705728000
Ok
```

Эта программа вычисляет $N!=1\cdot 2\cdot 3\ldots N$ для разных от 1 до N_{max}. Результат точный и может содержать до 200 знаков (их число можно изменить, изменив размерность массива A). Переменная W=52 задает максимальное число знаков при выводе в одной строке (если оно превышено, происходит перевод строки). Программа реализует обычный алгоритм вычисления ряда $1\cdot 2\cdot 3\ldots N$, но с применением символьных операций MID$, STR$ и CHR$. Это позволяет снять ограничения на число верных знаков результата. После листинга программы дан контрольный пример.

Программа П2.2

Точное умножение чисел с плавающей точкой (до 255 знаков в каждом числе)

```
10 PRINT "EXACT MULTIPLICATION FOR FLOATING-POINT NUMBERS"
20 W=60:DIM N(255),M(255),T(510):K=0
30 PRINT :INPUT "INPUT S(stop) OR NUMBER ";N$
40 IF N$="S" OR N$="s" THEN END
```

```
50 PRINT "MULTIPLIER":INPUT M$:A$=N$:GOSUB 210
60 N$=A$:L1=LEN(A$):D1=D:A$=M$:GOSUB 210
70 M$=A$:L2=LEN(A$):P=D1+D
80 FOR I=1 TO K:N(I)=0:M(I)=0:T(I)=0:NEXT I
90 FOR I=1 TO L1:N(L1-I+1)=VAL(MID$(N$,I,1)):NEXT I
100 FOR I=1 TO L2:M(L2-I+1)=VAL(MID$(M$,I,1)):NEXT I
110 PRINT "PRODUCT
120 FOR I=1 TO L2:FOR J=1 TO 510:K=J+I-1
130 T(K)=N(J)*M(I)+T(K)+C:C=INT(T(K)/10)
140 IF C=0 AND J>=L1 THEN 160
150 T(K)=T(K)-10*C:NEXT J
160 NEXT I:IF P>K THEN K=P
170 FOR J=K TO 1 STEP -1:IF J=P THEN PRINT ".";:W=W-1
180 PRINT MID$(STR$(T(J)),2);:KJ=K-J+1
190 IF KJ=W*INT((KJ)/W) THEN PRINT CHR$(13) CHR$(10) ;
200 NEXT J:PRINT:GOTO 30
210 L=LEN(A$):FOR I=L TO 1 STEP -1
220 IF MID$(A$,I,1)="." THEN 240
230 NEXT I:D=0:RETURN
240 D=L-I:IF I>1 THEN A$=MID$(A$,1,I-1)+MID$(A$,I+1):RETURN
250 FOR I=2 TO L:IF MID$(A$,I,1)<>"0" GOTO 270
260 NEXT I
270 A$=MID$(A$,I):RETURN

RUN
EXACT MULTIPLICATION FOR FLOATING-POINT NUMBERS

INPUT S(stop) OR NUMBER ? 123.4567890987654321
MULTIPLIER
? 987.65432123456789
PRODUCT
121932.631239140373304069513112635269

INPUT S(stop) OR NUMBER ? S
Ok
```

Здесь также реализован обычный алгоритм умножения чисел, но с применением символьных операций. Число верных знаков результата может доходить до 510. Под листингом программы дан контрольный пример.

Программа П2.3

Построение графика функции и координатных осей

```
10 SCREEN 2:CLS:KEY OFF
20 PSET (10,65)
30 FOR X=0 TO 600 STEP 2
40   A=60*EXP(-X/200)
50   Y=-A*SIN(X/4)
60   LINE -(X+10,Y+65)
70 NEXT
80 LINE (10,65)-(610,65)
90 LINE (10,5)-(10,125)
100 FOR Y=5 TO 125 STEP 12
110   LINE (10,Y)-(13,Y)
120 NEXT
130 LOCATE 1: PRINT"1.0"
140 LOCATE 9: PRINT"0"
150 LOCATE 17: PRINT"-1.0"
```

```
160 FOR X=70 TO 610 STEP 60
170   LINE (X,65)-(X,68)
180   LINE (X+1,65)-(X+1,68)
190 NEXT
200 LOCATE 10,38: PRINT "1.0"
210 LOCATE 10,76: PRINT "2.0"
220 A$=INPUT$(1)
```

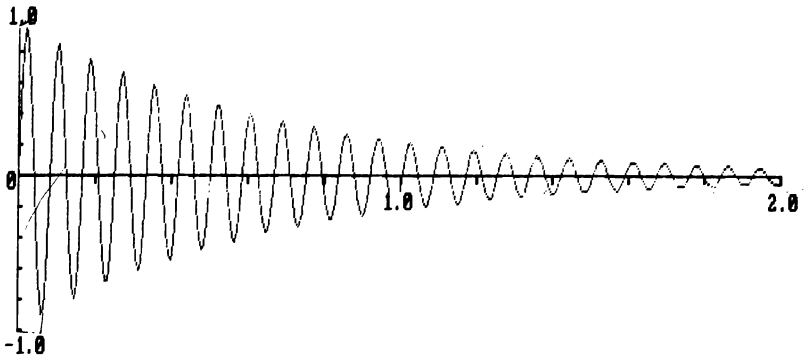
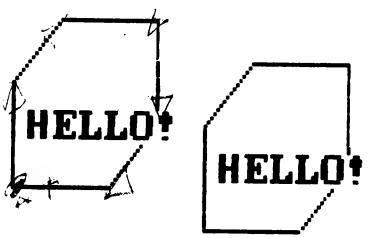


График функции (строки 40, 50) строится с помощью операторов PSET (намечается первая точка) и LINE. В строке 10 задан режим высокого разрешения SCREEN 2 (640×200 точек), очистка экрана CLS и очистка служебной строки с подсказками (операторы KEY OFF). Координатные оси строятся с помощью оператора LINE (строки 80–220), а поясняющие надписи – с помощью оператора LOCATE (строки 130–150, 200, 210). Строка 220 обеспечивает останов без "загрязнения" экрана. Для печати графиков нужно нажать одновременно клавиши Shift и Prtsc (Print Screen – печать экрана, это аналог команды COPY). При работе с графикой нужно ввести файл graphics ДОС. Полученный график приведен под программой.

Программа П2.4

Действие операторов GET и PUT

```
10 SCREEN 1: CLS: PSET (20,50)
20 HS$="U25E15R30D25G15L30"
30 DRAW HS$: LOCATE 5,4: PRINT "HELLO!"
40 DIM F(628)
50 GET (10,10)-(80,60),F
60 PUT (70,20),F
70 A$=INPUT$(1)
```

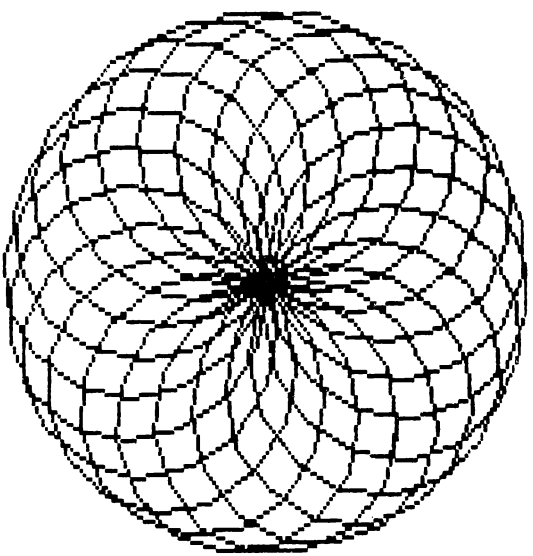


В строке 20 задана цепочка команд для оператора DRAW в виде значения символьной переменной HS$. Этот оператор строит шестиугольник (строка 30), а затем в него помещается надпись HELLO! (см. рисунок под программой). В строке 40 резервируется массив F в ОЗУ под хранение изображенного рисунка. Оператор GET в строке 50 снимает копию с рисунка и засылает ее цифровые коды в массив F. Наконец, оператор PUT (строка 60) выводит копию рисунка (она показана справа) на новое место экрана. Меняя координаты этого места, можно перемещать рисунок по экрану (т.е. возможна и динамическая графика).

Программа П2.5

Построение графика функции в полярной системе координат

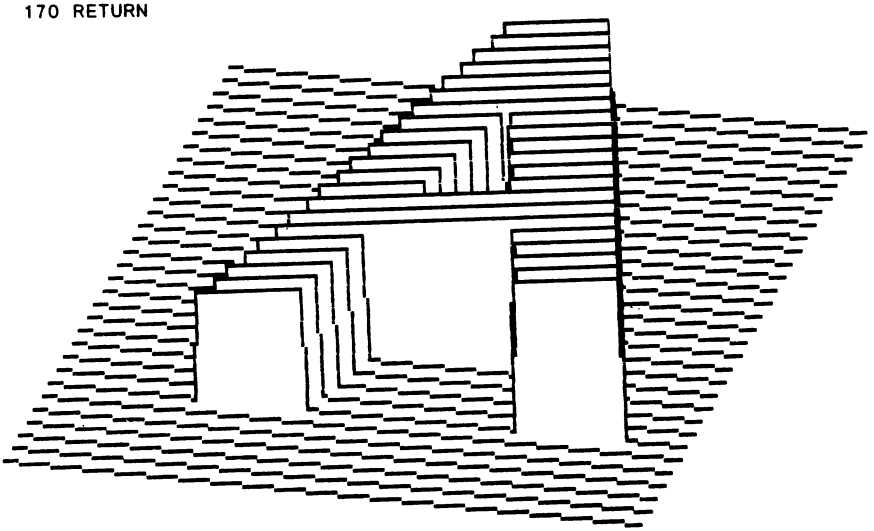
```
10 SCREEN 1: CLS
20 KEY OFF: T=80
30   FOR W=-40 TO 23.2 STEP .2
40     R=T*(SIN(W+F)*.8)
50     X=120+1.3*R*COS(W)
60     Y=100+R*SIN(W)
70     IF W=-40 THEN PSET(X,Y)
        ELSE LINE-(X,Y)
80   F=F+.02
90   NEXT W
100 A$=INPUT$(1)
```



Программа строит сложную кривую, которую описывает конец радиуса с переменной длиной R (строка 40). В строках 50 и 60 вычисляются проекции конца на оси X и Y. Изменение аргумента W зависимости R(W, F) задается в цикле, а F в строке 80.

Построение объемной буквы А, лежащей на плоскости

```
10 SCREEN 1:CLS:FA=3.333
20 FOR X=-20 TO 130 STEP 5
30   FOR Y=0 TO 200
40     GOSUB 120
50     NY=Y-X*.5+80:NZ=Z+X*.6+60
60     LINE(NY,NZ)-(NY,199),0
70     IF Y=0 THEN PSET(NY,NZ)
       ELSE LINE(PY,PZ)-(NY,NZ)
80     PY=NY:PZ=NZ
90   NEXT Y
100 NEXT X
110 M$=INPUT$(1):END
120 REM FUNCTION
130 Z=Y*.1:XT=X*.1:YT=(Y+120)*.06
140   IF XT<0 OR XT>10 THEN 170
150   IF XT<-FA*(YT-10)+10 OR XT<FA*(YT-10)-16.66666 THEN 170
160   IF XT<-FA*(YT-10)+16.66666 OR XT<FA*(YT-10)-10
      OR (XT>6 AND XT<8) THEN Z=-20
170 RETURN
```



Программа реализует алгоритм построения трехмерной фигуры удалением скрытых от наблюдателя линий. Фигура задается подпрограммой (строки 120–170). Два цикла задают значения координат X и Y, по ним вычисляется координата Z. Ограничения на координаты задаются видом функции (строки 130–160). Построения идут от заднего плана к переднему. Построение каждой новой линии (слева направо) идет по точкам. Одновременно ниже этой точки строится отрезок прямой до нижней границы экрана с цветом экрана. Это и обеспечивает стирание всех ранее построенных точек, если они заслоняются новыми точками. Этот алгоритм дает достаточно высокое качество построенных объемных фигур, но действует довольно медленно.

СЛОВАРЬ КОМАНД РАЗЛИЧНЫХ ВЕРСИЙ ЯЗЫКА БЕЙСИК

Ниже приводятся основные команды (свыше 200) наиболее распространенных версий языка Бейсик отечественных и зарубежных ПЭВМ. Для облегчения перевода программ с одной версии Бейсика на другую приведены данные о встречаемости команд в ряде типов ДОС и конкретных ПЭВМ (наличие команды в составе версии отмечено знаком +, отсутствие –). Символы $ и ¤ считаются равноценными. ДОС и РАФОС применяются в ПЭВМ ДВК-2М и ДВК-3.

Команда	ДОС или тип ПЭВМ								Стр.
	БК-0010	ДОС РАФОС	Агат Apple-II	Д3-28	FX-702P	IBM PC	ZX-Spectrum	Aquarius	
ABS	+	+	+	+	+	+	+	+	37
AC	–	–	–	–	+	–	–	–	149
ACS	–	+	+	+	+	+	+	–	35
AHC	–	–	–	+	+	–	–	–	36
AHS	–	–	–	+	+	–	–	–	36
AHT	–	–	–	+	+	–	–	–	36
AND	+	–	–	–	–	+	+	+	42
ASC	+	+	+	–	–	+	–	+	77
ASN	–	+	+	+	+	+	+	–	35
ASTAT	–	–	–	–	+	–	–	–	74
APPEND	–	+	+	–	–	–	–	–	166
AT	+	–	–	–	–	–	+	–	24
ATN	+	+	+	+	+	+	+	–	35
ATTR	–	–	–	–	–	–	+	–	136
AUTO	+	–	–	–	–	+	–	–	153
BEEP	+	–	–	–	–	+	+	–	145
BIN	–	+	–	–	–	+	+	–	128
BLOAD	+	–	+	–	–	+	–	–	166
BSAVE	+	–	+	–	–	+	–	–	166
BRIGHT	–	–	–	–	–	–	+	–	116
BRUN	–	–	+	–	–	+	–	–	166
BORDER	–	–	–	–	–	–	+	–	116
CALL	+	+	–	–	–	+	–	–	76
CAT	–	–	–	–	–	–	+	–	165
CATALOG	–	–	+	–	–	+	–	–	165
CINT	+	–	–	–	–	+	–	–	37
CIRCLE	+	–	–	–	–	+	+	–	125
CHAIN	–	+	+	–	–	+		–	166
CHR$	+	+	+	–	–	+	+	+	77
CLEAR	+	+	+	+	–	+	+	–	149
CLOAD	+	–	–	–	–	–	–	+	163

Команда	ДОС или тип ПЭВМ								Стр.
	БК-0010	ДОС РАФОС	Агат Apple-II	ДЗ-28	ГХ-702Р	IBM PC	ZX-Spectrum	Aquarius	
CLOSE	+	+	+	–	–	+	–	–	169
CRL	–	–	–	–	+	–	–	–	75
CLRALL	–	–	–	–	+	–	–	–	75
CLS	+	–	–	–	–	+	+	–	56
CODE	–	–	–	–	–	–	+	–	57
COLOR	+	–	+	–	–	+	–	–	58
CONT	+	–	–	–	+	–	–	+	57
CONTINUE	–	–	–	–	–	+	+	+	162
CSAVE	+	–	–	–	–	–	–	+	112
COM	–	+	–	+	–	+	–	–	101
CON	–	–	–	+	–	–	–	–	101
COPY	–	–	–	–	–	+	+	+	102
COR	–	–	–	–	+	–	–	–	102
COS	+	+	+	+	+	+	+	+	102
DATA	+	+	+	+	–	+	+	+	165
DEF FN	+	+	+	+	–	+	+	–	164
DEG	–	–	–	+	+	–	–	–	74
DEL	+	+	–	–	–	–	–	–	74
DELETE	+	–	+	–	–	+	+	+	44
DIM	+	+	+	+	–	+	+	+	165
DMS	–	–	–	–	+	–	–	–	116
DRAW	+	–	–		–	+	+	–	42
DRAWI	–	–	–	–	–	+	–	–	102
EDIT	–	–	–	–	–	+	+	–	39
ELSE	+	–	–	–	–	+	–	–	40
END	+	–	+	+	+	+	+	+	40
ENTER	–	–	–	–	+	–	+	–	51
EOX	–	–	–	–	+	–	–	–	40
EOY	–	–	–	–	+	–	–	–	42
EXE	–	–	–	–	+	–	–	–	166
EXEC	–	–	+	–	–	–	–	–	76
EXP	+	+	+	+	+	+	+	+	116
EXT	–	–	–	+	–	–	–	–	132
FIELD	–	–	–	–	–	+	–	–	116
FLASH	–	–	+	–	–	+	+	–	23
FN	+	+	+	+	–	+	+	–	76
FOR	+	+	+	+	+	+	+	+	15
FRAC	–	–	–	–	+	–	–	–	147
FRE	+	–	+	–	–	–	–	+	120

Команда	ДОС или тип ПЭВМ								Стр.
	БК-0010	ДОС РАФОС	Агат Apple-II	ДЗ-28	FX-702P	IBM PC	ZX-Spectrum	Aquarius	
GET	–		+	–	+	+	–	–	141
GOTO	+	+	+	+	+	+	+	+	39
GOTO#	–	–	–	–	+	–	–	–	40
GOSUB	+	+	+	+	+	+	+	+	49
GOSUB#	–	–	–	–	+	–	–	–	50
GR	–	–	+	–		–	–		112
GRAF	–	–	–	–	–	–	+	–	129
HEX$	+	–	+	–	–	+	+		39
HCS	–	–	–	+	+	–	–	–	36
HSN	–	–	–	+	+		–	–	36
HTN	–	–		+	+	–	–		36
IDN	–	–	–	+	–	–	–	–	59
IF	+	+	+	+	+	+	+	+	40
IN	–	–	–	–	–	–	+	–	76
INC	–	–	–	–	+	–	–	–	151
INIT	–	–	+	–	–	–	–	–	37
INK	–	–		–	–	–	+	–	116
INKEY$	+	+	+	–	–	+	+	+	54
INPUT	+	+	+	+	+	+	+	+	25
INPUT LINE	–	–		–	–	–	+	–	96
INT	+	+	+	+	+	+	+	+	37
INV	–	–	–	+	–	–	–		59
INVERSE	–	–	+	–	–		+	–	116
KEY	+	+	–	–	+	+	–	–	54
KILL	–	–	+	–	–	+	–	–	167
LEFT$	–	–	+	–	–	+	–	+	100
LEN	+	+	+	–	+	+	+	+	99
LET	+	+	+	+	+	+	+	–	26
LINE	+	+	–	–	–	+	–	–	121
LINPUT	–	–		–	–	+	+	–	96
LIST	+	+	+	+	+	+	+	+	152
LLIST	+	+	–	–	–	+	+	+	152
LGT	–	–	–	+	–	–	–	–	32
LN	–	–		–	+	–	–	–	32
LOAD	+	+	+	+	+	+	+		161
LOAD DATA	–	–	–	–	–	–	+	–	161
LOAD CODE	–	–	–	–	–	–	+	–	161
LOCK	–	–	+	–	–	–	–	–	165
LOG	+	+	+	+	+	+	+	+	33
LOG 1Ø	–	+	–		–	–	–	–	32

Продолжение табл.

Команда	ДОС или тип ПЭВМ								Стр.
	БК-0010	ДОС РАФОС	Агат Apple-II	ДЗ-28	FX-702P	IBM PC	ZX-Spectrum	Aquarius	
LPRINT	+	–	–	–	–	–	+	+	169
LRA	–	–	–	–	+	–	–	–	75
LRB	–	–	–	–	+	–	–	–	75
MAT	–	–	–	+	–	–	–	–	56
MAT INPUT	–	–	–	+	–	–	–	–	57
MAT PRINT	–	–	–	+	–	–	–	–	58
MAT READ	–	–	–	+	–	–	–	–	57
MERGE	–	–	–	–	–	+	+	–	162
MGR	–	–	+	–	–	–	–	–	112
MID	–	–	–	–	+	–	–	–	101
MID$	+	+	+	–	–	+	–	+	101
MKD¤	–	+	–	–	–	+	–	–	102
MKI¤	–	+	–	–	–	+	–	–	102
MKS¤	–	+	–	–	–	+	–	–	102
MON	–	–	+	–	–	–	–	–	165
MOTOR	–	–	–	–	–	+	–	–	164
MY	–	–	–	–	+	–	–	–	74
MX	–	–	–	–	+	–	–	–	74
NEW	+	+	+	–	–	+	+	–	74
NEXT	+	+	+	+	+	+	+	+	44
NOMON	–	–	+	–	–	–	–	–	165
NORMAL	–	–	+	–	–	–	+	–	116
NOT	+	–	–	–	–	–	+	–	42
OCT	+	+	–	–	–	+	–	–	102
OCT¤	–	–	–	–	–	–	–	–	39
ON	+	+	+	+	–	+	–	+	40
ON GOTO	+	+	+	+	–	+	–	+	40
ON GOSUB	+	+	+	+	–	+	–	+	51
ON THEN	+	+	+	+	–	+	–	+	40
OR	+	–	–	–	+	+	+	–	42
OPEN	+	+	+	–	–	+	+	–	166
OUT	+	–	–	–	–	–	+	–	76
OVER	–	–	–	–	–	–	+	–	116
PAINT	+	–	–	–	–	+	–	–	132
PAPER	–	–	–	–	–	–	+	–	116
PAUSE	–	–	–	+	–	+	+	–	23
PEEK	+	–	+	–	–	+	+	+	76
PI(π)	+	+	+	+	+	+	+	–	15
PLAY	–	–	–	–	–	+	–	–	147
PLOT	–	–	–	–	–	+	+	–	120

Продолжение табл.

Команда	ДОС или тип ПЭВМ								Стр.
	БК-0010	ДОС РАФОС	Агат Apple-II	Д3-28	FX-702P	IBM PC	ZX-Spectrum	Aquarius	
PLOT–TO	–	–	+	–	–	–	–	–	120
POINT	+	–	–	–	–	+	+	+	119
POKE	+	–	+	–	–	+	+	+	76
POS	+	+	+	–	–	+	–	+	153
POSITION	–	–	+	–	–		–	–	167
PRC	–	–	–	–	+	–	–	–	38
PRESET	+	–	–	–	–	+	–	+	118
PRINT	+	+	+	+	+	+	+	+	20
PSET	+	–	–	–	–	+	–	+	118
PUT	–	–	+	–	–	+	–	+	141
RAD	–	–	–	+	+	–	–	–	35
RANDOMIZE	–	+	+	+	–	+	+	–	55
RAN#	–	–	–	–	+	–	–	–	55
READ	+	+	+	+	–	+	+	+	27
RENUM	+	+	+	–	–	+	–	–	154
REM	+	+	+	+	–	+	+	+	20
RESTORE	+	+	+	+	–	+	+	+	27
RET	–	–	–	–	+	–	–	–	51
RETURN	+	+	+	+	–	+	+	+	49
RIGHT$	–	–	+	–	–	–	–	+	101
RND	+	+	+	+	+	+	+	+	55
ROT	–	–	+	–	–	–	–	–	130
RPC	–	–	–	–	+	–	–	–	38
RUN	+	+	+	+	+	+	+	+	149
SAC	–	–	–	–	+	–	–	–	149
SAVE	+	+	+	+	+	+	+	–	160
SAVE ALL	–	–	–	–	+	–	–	–	160
SAVE DATA	–	–	–	–	–	–	+	–	160
SAVE CODE	–	–	–	–	–	–	+	–	160
SAVE END	–	–	–	+	–	–	–	–	160
SAVE LINE	–	–	–	–	–	–	+	–	160
SAVE#	–	–	–	–	+	–	–	–	160
SET	–	–	–	–	+	–	–	–	23
SCALE	–	–	+	–	–	–	–	–	130
SCREEN$	–	–	–	–	–	+	+	–	136
SCRN	–	–	+	–	–	–	–	–	119
SDY	–	–	–	–	+	–	–	–	74
SDX	–	–	–	–	+	–	–	–	74
SDXY	–	–	–	–	+	–	–	–	74
SGN	+	+	+	+	+	+	+	+	37

Команда	ДОС или тип ПЭВМ								Стр.
	БК-0010	ДОС РАФОС	Агат Apple-II	Д3-28	FX-702P	IBM PC	ZX-Spectrum	Aquarius	
SIN	+	+	+	+	+	+	+	+	34
SQR	+	+	+	+	+	+	+	+	33
STOP	+	+	+	–	+	+	+	+	149
SOUND	–	–	–	–	–	+	–	+	145
STAT	–	–	–	–	+	–	–	–	74
STEP	+	+	+	+	+	+	+	+	44
STR$	+	+	+	–	–	+	+	+	99
STRING$	+	–	–	–	–	–	–	–	103
SY	–	–	–	–	+	–	–	–	74
SY2	–	–	–	–	+	–	–	–	74
SX	–	–	–	–	+	–	–	–	74
SX2	–	–	–	–	+	–	–	–	74
SXY	–	–	–	–	+	–	–	–	74
SWAP	–	–	–	–	–	+	–	–	27
TAB	+	+	+	+	–	+	+	+	23
TAN	+	+	+	+	+	+	+	+	34
TEXT	–	–	+	–	–	–	–	–	112
THEN	+	+	+	+	+	+	+	+	41
TO	+	+	+	+	+	+	+	+	44
TRACE	+	–	–	–	+	–	–	–	154
TRM¤	–	+	–	–	–	+	–	–	102
TRN	–	–	–	+	–	–	–	–	59
UNTIL	–	–	–	–	–	+	–	–	49
USR	+	–	+	–	–	+	+	+	78
VAL	+	+	+	–	–	+	+	+	100
VERIFY	–	–	–	–	–	+	+	–	163
WAIT	–	–	+	–	+	–	–	–	23
WHILE	–	–	–	–	–	+	–	–	49
WIDHT	–	–	–	–	–	+	–	–	154
WINDOW	–	–	–	–	–	+	–	–	133
WRITE	–	–	+	–	–	+	–	–	166
ZER	–	–	–	+	–	–	–	–	58

СПИСОК ЛИТЕРАТУРЫ

1. **Дьяконов В. П.** Справочник по алгоритмам и программам на языке Бейсик для персональных ЭВМ. – М.: Наука, 1987. – 240 с.
2. **Громов Г. Р.** Национальные информационные ресурсы: проблемы промышленной эксплуатации. – М.: Наука, 1984. – 237 с.
3. **Дьяконов В. П.** Справочник по расчетам на микрокалькуляторах. – М.: Наука, 1986. – 224 с.
4. **Стукалов П. М.** Производство товаров для населения – на уровень новых задач// Электронная промышленность. – 1986. – № 7. – С. 77.
5. **Иоффе А. Ф.** Массовые персональные ЭВМ типа "Агат"//Микропроцессорные средства и системы. – 1984. – № 1. – С.56.
6. **Трейстер Р.** Персональный компьютер фирмы IBM. – М.: Мир, 1986. – 208 с.
7. **Кейтер Дж.** Компьютеры-синтезаторы речи. – М.: Мир, 1985. – 235 с.
8. **Дьяконов В. П.** Персональные ЭВМ в аппаратуре и технике эксперимента//Приборы и техника эксперимента. – 1986. – № 1. – С. 7.
9. **Яблонский Ф. М., Троицкий Ю. В.** Средства отображения информации. – М.: Высшая школа, 1985. – 200 с.
10. **Барни К.** Macintosh – новый личный компьютер с развитыми функциональными возможностями//Электроника. – 1984. – № 2. – С. 65.
11. **Бытовая** персональная микро-ЭВМ "Электроника БК-0010"/С. М. Косенков, А. П. Полосин, З. А. Счепицкий и др.//Микропроцессорные средства и системы. – 1985. – № 1. – С. 22.
12. **Барни К.** Карманный компьютер, совмещающий функции калькулятора и контроллера//Электроника. – 1984. – № 1. – С. 113.
13. **Попов А. А., Хохлов М. М., Глухман В. Л.** Диалоговые вычислительные комплексы "Электроника НЦ-80-20"//Микропроцессорные средства и системы. – 1984. – № 4. – С. 61.
14. **Сергеева Т. О.** HP-85 настольная ЭВМ индивидуального пользования//Зарубежная радиоэлектроника. – 1981. – № 10. – С. 86.
15. **Котшенрутер К., Энгельс В. А. Х.** Личный компьютер-аналог мини-компьютера// Электроника. – 1982. № 5. – С. 38.
16. **Кетков Ю. Л.** Программирование на БЕЙСИКе. – М.: Статистика, 1978. – 158 с.
17. **Уорт Т.** Программирование на языке БЕЙСИК. – М.: Машиностроение. – 1981. – 225 с.
18. **Программирование** на языке БЕЙСИК-ПЛЮС для СМ-4/В. П. Семик, Б. Р. Мончибович, О. П. Непочатых и др. – М.: Финансы и статистика, 1982. – 246 с.
19. **Александров А. Л., Башмакова Е. С., Гуткин А. Б., Либеров А. Б.** О стандарте языка БЕЙСИК//Микропроцессорные средства и системы. – 1986. – № 5. – С. 25.
20. **Корчак А. Е.** Языки программирования БЕЙСИК для микро-ЭВМ. – М. : МЦНТИ, МНИИПУ, 1987.
21. **Безродный М. С.** Классификация дисплеев для микро-ЭВМ//Микропроцессорные средства и системы. – 1986. – № 4. – С. 28.

22. **Погорелый С. Д., Слободянюк Т. Ф.** Программное обеспечение микропроцессорных систем: Справочник. – Киев: Техника, 1985. – 236 с.
23. **Глушкова Г. Г., Иванов Е. А.** Микро-ЭВМ семейства "Электроника"//Микропроцессорные средства и системы. – 1986. № 4. – С. 7.
24. **Hofman P.** The MSX Book. – M. Graw Hill Company. – 292 c.
25. **Programmable** Calculator Casio FX-702P. Instruction Manuil. Japan Busines Machine Makers Association. 1982. – 49 p.
26. Von Steven VicKers. Sinclair ZX-Spectrum BASIC-Programmerung. Herstellung. Werbeagentur Cooperation. – Miinchen, 1984. – 230 p.
27. **Schneider D. I.** Handbook of BASIC for the IBM personal Computer. – Bowie: Brady, 1983.
28. **Прохоров Е. М.** Рынок персональных микро-ЭВМ в ФРГ. М., 1985. – 20 с. (Экспресс-информация. Приборы, средства автоматизации и системы управления/ ЦНИИТЭИ. Вып. 1).
29. **Вычислительная** математика/Н. И. Данилина, Н. С. Дубровская, О. П. Кваша, Г. Л. Смирнова. – М.: Высшая школа, 1965. – 472 с.
30. **Волков Е. А.** Численные методы. – М.: Наука, 1982. – 256 с.
31. **Самарский А. А.** Введение в численные методы. – М.: Наука. – 1982. – 272 с.
32. **Справочник** по специальным функциям/Под ред. М. Абрамовица и И. Стигана. – М.: Наука, 1979. – 832 с.
33. **Бронштейн И. Н., Семендяев К. А.** Справочник по математике для инженеров и учащихся вузов. – М.: Наука, 1980. – 976 с.
34. **Энджел Й.** Практическое введение в машинную графику: Пер. с англ./Под ред. В. А. Львова. – М.: Радио и связь, 1984. – 136 с.
35. **Егоров М. И., Патрикеев Ю. Н., Ясеневский С. В.** Машинная графика в ОС ЕС. – М.: Финансы и статистика, 1984. – 159 с.
36. **Гилой В.** Интерактивная машинная графика: Структуры данных, алгоритмы, языки/Пер. с англ. под ред. Ю. М. Баяковского – М.: Мир, 1981. – 384 с.
37. **Ньюмен У., Спрулл Р.** Основы интерактивной машинной графики/Пер. с англ. под ред. В. А. Львова. – М.: Мир, 1976. – 574 с.
38. **Роджерс Д., Адамс Дж.** Математические основы машинной графики/Пер. с англ. под ред. Ю. И. Топчеева. – М.: Машиностроение, 1980. – 240 с.
39. **Расчет** электрических цепей и электромагнитных полей на ЭВМ/М. Г. Александрова, А. Н. Белянин, В. Брюкнер и др.; Под ред. Л. В. Данилова и Е. С. Филиппова. – М.: Радио и связь, 1983. – 344 с.
40. **Батищев Д. И.** Методы оптимального проектирования. – М.: Радио и связь, 1984. – 248 с.
41. **Гилл Ф., Мюррей У., Райт М.** Практическая оптимизация: Пер. с англ. – М.: Мир, 1985. – 509 с.
42. **Чуа Л. О., Пен-Мин Лин.** Машинный анализ электронных схем: Пер. с англ./Под ред. В. Н. Ильина. – М.: Энергия, 1980. – 640 с.
43. **Ильин В. Н.** Основы автоматизации схемотехнического проектирования. – М.: Энергия, 1979. – 392 с.
44. **Зернов Н. В., Карпов В. Г.** Теория радиотехнических цепей. – Л.: Энергия, 1972. – 816 с.
45. **Кочетков Г. Б.** Компьютерные игры: свет и тени//Микропроцессорные средства и системы. – 1985. – № 3. – С. 16.
46. **Основы** информатики и вычислительной техники: Учебное пособие для 9-го класса средней школы/Под ред. А. П. Ершова и В. М. Монахова. – М.: Просвещение, 1985.

47. **Громов Г. Р.** Профессиональные приложения персональных ЭВМ//Микропроцессорные средства и системы. – 1985. № 3. – С. 9.
48. **Dossett R. J.** Perconal Computers: What They Can Do for IEs and How to Choose One// Industrial Eng. – 1982. – V. 14, № 10. – P. 70.
49. **Филдс Ст. У.** Личный компьютер становится рабочим инструментом инженера// Электроника. – 1983. № 24. – С. 91.
50. **Олстер Н.** Развитие рынка контрольно-измерительной аппаратуры, выполненной на основе личных компьютеров//Электроника. – 1983. – № 6. – С. 80.
51. **Уоллер Л.** Личные компьютеры в качестве измерительных приборов//Электроника. – 1983. – № 4. – С. 3.
52. **Berk A. A.** QL SUPER BASIC. – Granada: London, Toronto, Sydney, New York, 1984, 165 p.
53. **Пул Л.** Работа на персональном компьютере: Пер. с англ./Под ред. Е. К. Масловского. – М.: Мир, 1986. – 383 с.
54. **Beta BASIC.** Release 3.0. – London: Betasoft, 1984. – 88 p.
55. **Mega** BASIC. Version 4.0. – London: Mike Leaman, 1985. – (на кассете).
56. **Sprite** BASIC. – Krakow: Mapasoft, 1986. – (на кассете).
57. **Моррил Г.** Бейсик для ПК IBM: Пер. с англ./Под ред. С. В. Черемных. – М.: Финансы и статистика, 1987. – 207 с.
58. **Уолш Б.** Программирование на Бейсике: Пер. с англ. – М.: Радио и связь, 1988. – 336 с.

ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

Аксонометрические изображения 188
Анализ машинный 224
– – лестничных цепей 226, 256
– – спектральный 242, 264
– – схем произвольной конфигурации 230
Аппроксимация функций 64, 182
– – двух переменных 213
– – методом наименьших квадратов 219
– – многоинтервальная 183
– – обратных тригонометрических 73
– – полиномиальная 64, 183
– – сплайнами 183
– – специальных 177
– – чебышевская 219

Быстрое преобразование Фурье (БПФ) 244, 266

Ввод и вывод 20, 25

Гамма-функция 177
Гистограммы линейные 186, 222
– круговые 222
Горнера схема 65
Графемы 127
Графика динамическая 138, 194
– трехмерная 193, 273
Графики атрибуты 112, 130
Графики на плоскости 181
Графические операторы 114, 130
Графопостроитель 7

Диалога организация 109
Дискретное преобразование Фурье (ДПФ) 245
Директивы 14
Дисплей 6, 128
Дифференцирование функций 51

Задание окон 133
– углов 35, 38
– цвета 115, 116
Закраска фигур 132, 202
Запись данных и программ 161
Запоминающее устройство 5
Звуковые эффекты 145

Интегрирование численное 178, 246
Интерполяция 64
Интерпретатор 12
Интерфейс 5

Калейдоскоп 205
Калькулятора режим 30
Кодов ввод, вывод, преобразование 76
Команды 14, 274
Комментарии 20
Компиляция 12, 158
Комплексные числа 67
Константы числовые 16
– символьные 95
Контроль тестовый 170
Координат преобразование 38, 196
Копирование экрана 109
Курсор 82

Лиссажу фигуры 195
Листинг программы 152
Логографика 145

Магнитофон кассетный 6, 159
Массивы 18
Матрицы 58
Микропроцессор 5
Модели дискретные 224, 233
Мозаика 205
Монитор 12
Мышь 7, 208

Накопитель на магнитных дисках 6, 164

Оператор 15
– присваивания 28, 62, 86
Операции арифметические 28, 61
– с константами 98, 104
– матричные 57, 196
– с полиномами 179, 218
– с символьными переменными 98, 104
– сравнения 40, 106
Операционная система 164
Отладка программ 157
Ошибок индикации 156

Перенумерация строк 154

Переменные глобальные 18, 19, 83
– комплексные 67
– локальные 83
– символьные 95, 96, 99, 104
– системные 81
– целочисленные 18
Переменных состояния метод 238
Переходы безусловные 39
– условные 40
Периферийное оборудование 5
Перо световое 208
Пиксель 111
Подпрограмма 49
Полином 64
Покрытие фигур 202
Порты 76
Построение дуг 122
– окружностей 125
– прямых 119
– рисунков на плоскости 186
– точек 181
– эллипса 123
Представление чисел 17
Принтер 6
Приоритет операций 30
Проверка программ 163
Программа линейная 13, 15
– разветвляющаяся 41
– циклическая 44
Программирование структурное 82
Процедура 82
Пульт клавишный 5

Разрешение графики 111
Регрессия 220, 254
– линейная 74
Редактирование переменных 93
– программ 150
Редакторы графические 208, 267
Рекуррентные выражения 176
Рекурсия 85
Ремонт ПЭВМ 171

Синус интегральный 176
Системы уравнений линейных 62
– – дифференциальных 179, 248
– – – с комплексными коэффициентами 68
– нелинейных 65, 178, 246
Случайные числа 55
Сортировка чисел 92
символов 108
Спектральный анализ последовательный 242
– – с выводом графиков 242
Сплайн-аппроксимация 183
Спрайты 140
Статистические расчеты 74
Стирлинга формула 177
Страницы графические 111
текстовые 112
– смешанные 112

Табуляция 23
Типовые программы графики 218
Трассировка 154

Управление дисплеем 168
– накопителем кассетным 160
– – дисковым 165
– принтером 169
– ПЭВМ 149
– шрифтом 137
Усечение фигур 201

Файлы 162, 165
Факториал 176
Формат вывода 23
– чисел 17
Функции алгебраические 15, 31
– гиперболические 36
– логические 89
– комплексной переменной 175
– математические 173
– пользователя 52
– тригонометрические 34
– символьных переменных 99
– специальные 176
– числовые 37

Характеристики ПЭВМ 7

Циклы 43

Чисел представление 16
Числа двоичные 12
– десятичные 16
– шестнадцатеричные 12

Языки программирования 12
Якоби матрица 65

ОГЛАВЛЕНИЕ

Предисловие 3

Глава 1. Общие сведения о персональных ЭВМ 4

1.1. Назначение. 4
1.2. Структура и состав оборудования 5
1.3. Технические характеристики 7
1.4. Программное обеспечение ПЭВМ 10

Глава 2. Основы программирования расчетных операций на языке Бейсик 13

2.1. Алфавит языка Бейсик, структура программ и виды переменных 13
2.2. Вывод информации (операторы REM, PRINT, SET, TAB, AT, WAIT, PAUSE). 20
2.3. Ввод числовых данных (операторы INPUT, LET, DATA, READ, RESTORE, SWAP) 25
2.4. Арифметические операции, работа в режиме калькулятора. 28
2.5. Вычисление элементарных алгебраических функций (EXP, EXT, LN, LOG, LGT, SQR) 31
2.6. Вычисление тригонометрических и обратных тригонометрических функций (SIN, COS, TAN, ASN, ACS, ATN, PI, RAD, DEG). 34
2.7. Вычисление гиперболических и обратных гиперболических функций (HSN, HCS, HTN, AHS, AHC, AHT). 36
2.8. Основные числовые и дополнительные функции (ABS, SGN, INT, FRAC, RPC, DMS, RND, HEX$, OCT$) 37
2.9. Безусловные и условные переходы (операторы GO TO, ON – GOTO, IF, THEN, AND, OR, ELSE, UNLESS). 39
2.10. Организация циклов (операторы FOR, TO, STEP, NEXT, WHILE, UNTIL) 43
2.11. Организация подпрограмм (операторы GOSUB, RETURN, RET, ON – GOSUB) 49
2.12. Организация функций пользователя (DEF FN, INKEY, KEY). 52
2.13. Генерация случайных чисел (операторы RND, RAN#, RANDOMIZE). . . . 55

Глава 3. Программирование специальных расчетных операций на ПЭВМ 56

3.1. Особенности программирования с применением матричных операций (операторы MAT READ, MAT INPUT, MAT PRINT, ZER, CON, IDN, TRN, INV) 56
3.2. Операции, функции и системы линейных уравнений с комплексными переменными 67
3.3. Особенности расчетов на ПЭВМ с малой емкостью ОЗУ (оператор DEFM) 71
3.4. Статистические расчеты на специализированных ПЭВМ (операторы STAT, ASTAT, SAC, функции SDX, SDY, SDXN, SDYN, LRA, LRB, COR, EOX, EOY, MX, MY, SX, SY, SX2, SY2, SXY) 74
3.5. Работа в кодах (операторы OUT, POKE, USR, CALL, функции IN, CODE, PEEK, CHR$, ASC) 76

3.6. Структурное программирование (операторы DEF PROC, END PROC, LOCAL, DEFAULT, DO, LOOP, REPEAT) 82
3.7. Дополнительные функции и операторы (AND, COSE, SINE, OR, RNDM, DEC, MOD, MEM, DPOKE, DPEEK, DEF KEY, JOIN, KEYWORD, SORT, ITEM, XOR) 89

Глава 4. Программирование операций с символьными переменными 95

4.1. Символьные константы и переменные 95
4.2. Ввод и вывод символьных переменных и организация их массивов (операторы INPUT, LET, INPUT LINE, DATA – READ – RESTORE, PRINT, TAB, AT) 96
4.3. Функции символьных переменных (LEN, STR$, VAL, VAL$, LEFT$, RIGHT$, MID$, TO и др.) 99
4.4. Операции с символьными переменными и константами 104
4.5. Применение символьных переменных для организации диалога с ПЭВМ. . 109

Глава 5. Программирование графики и звуковых эффектов 111

5.1. Задание графических и текстовых страниц (директивы GR, MGR, TEXT). 111
5.2. Задание цвета (операторы COLOR, INK, PAPER, BORDER, BRIGHT, FLASH, OVER, NORMAL, INVERSE) 114
5.3. Построение точек (операторы PSET, PRESET, PLOT, функции POINT и SCRN) 117
5.4. Построение отрезков прямых и дуг (операторы PLOT – TO, DRAW, LINE) 119
5.5. Построение эллипсов и окружностей (оператор CIRCLE) 123
5.6. Вывод и задание графем (операторы BIN, DRAWI, XDRAWI, SCALE, директива GRAF) 127
5.7. Дополнительные графические операторы и функции расширенных версий языка Бейсик (ALTER, BOX, DRAW TO, CIRCLE, PAINT, FILL, WINDOW, OPEN, CLOSE, BLOCK, POINT, CURSOR, ARC, XOS, YOS, XRG, YRG, SCRN$) 130
5.8. Управление шрифтом (операторы CSZE, SHIFT$, MODE, CHR$, FONT) . . 136
5.9. Динамическая графика (операторы ROLL, SCROLL, PAN, FADE, GET, PUT, PLOT, *GRAPHIC, *CHAR, *SPRITE и др.) 138
5.10. Логографика (операторы RENDOWN, PENUP, MOVE, TURN, TURN TO) 143
5.11. Задание звуковых эффектов (операторы BEEP, SOUND, PLAY, SREP, SON, SOFF, *ZAP, *NOISE) 145

Глава 6. Управление ПЭВМ и периферийным оборудованием, эксплуатация их . . 148

6.1. Управление ПЭВМ (директивы NEW, RUN, STOP, CONTINUE, END, CLR, CLRALL, VAC, SAC) 148
6.2. Ввод программы, вывод листинга, редактирование и трассировка (директивы HOME, LIST, LLIST, EDIT, CLEAR, DELETE, INC, AUTO, RENUM, TRACE, CLS, SCR, WIDTH) 150
6.3. Управление кассетным накопителем (директивы SAVE, PUT, LOAD, GET, MERGE, VERIFY, MOTOR, SCREEN$) 159
6.4. Управление накопителем на магнитных дисках (директивы DIR, CATALOG, CAT, UNIT, RENAME, DELETE, LOAD, SAVE, KILL и др.) 164
6.5. Подключение к ПЭВМ дисплея и принтера (директивы LLIST, LPRINT, COPY) 168
6.6. Модернизация и ремонт ПЭВМ 169

Глава 7. Некоторые применения персональных ЭВМ 172

7.1. Персональные ЭВМ в роли записной книжки. 172
7.2. Задание дополнительных математических функций 173
7.3. Математические расчеты (реализация основных численных методов). 178
7.4. Построение произвольных функций и их аппроксимация 181
7.5. Построение рисунков на плоскости и в аксонометрии 186
7.6. Построение движущихся объектов 194
7.7. Усечение, покрытие и специальная закраска фигур 201
7.8. Создание мозаичных изображений и калейдоскопов 205
7.9. Разработка графических редакторов-программ для построения и редактирования произвольных графиков 208
7.10. Аппроксимация и графическое представление на плоскости функций двух переменных. 213
7.11. Работа с типовыми программами графики 218
7.12. Анализ и моделирование электронных схем 224
7.13. Спектральный анализ сигналов 241

Приложение 1. Пакет прикладных программ 246
Приложение 2. Пакет демонстрационных программ для персональных ЭВМ класса IBM PC 268
Приложение 3. Словарь команд различных версий языка Бейсик 274

Список литературы 280
Предметный указатель 283

Производственное издание

ДЬЯКОНОВ ВЛАДИМИР ПАВЛОВИЧ

ПРИМЕНЕНИЕ ПЕРСОНАЛЬНЫХ ЭВМ И ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ БЕЙСИК

Заведующая редакцией **Г. И. Козырева**
Редактор **Т. М. Толмачева**
Художественный редактор **А. В. Проценко**
Переплет художника **В. Н. Забайрова**
Технический редактор **Т. Г. Родина**
Корректор **Т. В. Дземидович**

ИБ № 1971

Подписано в печать с оригинала-макета 29.03.89. Т-09902. Формат 60x88/16. Бумага офс. № 2. Гарнитура "Пресс-роман". Печать офсетная. Усл. печ. л. 17,64. Усл. кр.-отт. 17,64. Уч.-изд. л. 17,92. Тираж 40 000 экз. (1-й завод: 1 — 25 000 экз.). Изд. № 22620. Заказ № 6694. Цена 1 р. 20 к.
Издательство "Радио и связь". 101000 Москва, Почтамт, а/я 693

Ордена Октябрьской Революции и ордена Трудового Красного Знамени МПО "Первая Образцовая типография" Союзполиграфпрома при Государственном комитете СССР по делам издательств, полиграфии и книжной торговли. 113054 Москва, Валовая, 28